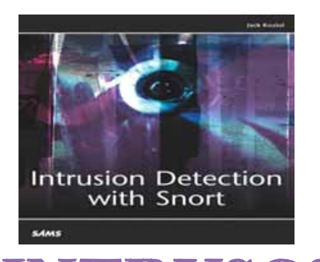
HACK X CRACK: HACKEANDO LAS RESTRICCIONES DE SEGURIDAD DE WINDOWS



IDS

SISTEMA DE DETECCION DE INTRUSOS



DESCUBRE LA VERDADERA SEGURIDAD

3 SERVIDORES ON LINE PARA TUS PRACTICAS DE HACK



LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

PROGRAMANDO SOCKETS EN PHP

HACKEANDO EL CORAZON DE WINDOWS

LAS DIRECTIVAS DE SEGURIDAD

Y

TE PRESENTAMOS SNORT

JUEGA GRATIS ON LINE CON LA XBOX

LINUX: COLAS DE MENSAJES



regedit.exe: Lo Crackeamos!!!

W32Dasm: EL DESENSAMBLADOR

CODIGO MAQUINA

INGENIERIA INVERSA

UltraEdit-32: EL EDITOR HEXADECIMAL

Nº 19 -- P.V.P. 4,5 EUROS

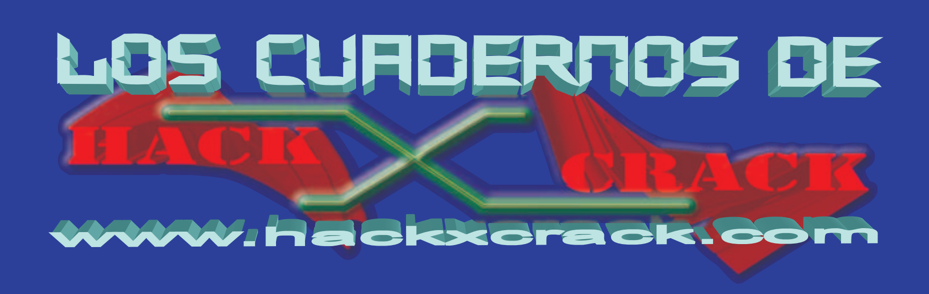


HACK X CRACK - HACK X CRACK - HACK X CRACK

LOS MEJORES ARTÍCULOS GRATIS EN NUESTRA WEB

PC PASO A PASO: APRENDE A PROGRAMAR SOCKETS !!!





EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701
PERE MARTELL Nº 20, 2º - 1ª
43001 TARRAGONA (ESPAÑA)

Director Editorial
I. SENTIS
E-mail contacto
director@editotrans.com
Título de la publicación
Los Cuadernos de HACK X CRACK.
Nombre Comercial de la publicación
PC PASO A PASO
Web: www.hackxcrack.com
Dirección: PERE MARTELL Nº 20, 2º - 1ª.
43001 TARRAGONA (ESPAÑA)



Director de la Publicación
J. Sentís

E-mail contacto
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

E-mail contacto:
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania, de Japón y algún Estadounidense.

E-mail contacto
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

DISTRIBUCIÓN:
SGEL, Avda. Valdeparra 29 (Pol. Ind.)
28018 ALCOBENDAS (MADRID)
Tel 91 657 69 00 FAX 91 657 69 28
WEB: www.sgel.es

TELÉFONO DE ATENCIÓN AL CLIENTE: 977 22 45 80
Petición de Números atrasados y Suscripciones (Srta. Genoveva)

HORARIO DE ATENCIÓN: DE 9:30 A 13:30
(LUNES A VIERNES)

NUMERO 19 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL
Deposito legal: B.26805-2002
Código EAN: 8414090202756

EDITORIAL

NO HAY MAL QUE POR BIEN NO VENGA...

Hace poco publicamos un artículo que fue criticado de forma muy positiva por nuestros lectores: Hackear Windows en 40 segundos. La verdad, ese tipo de artículos pertenecen al tipo "hackear sin saber".

Pero los lectores mandan y por mucho que nosotros nos esforcemos en ENSEÑAR y vosotros en aprender, debemos reconocer que también hay que divertirse, así que este mes podrás leer otro de estos artículos. Esta vez crackearemos el regedit.exe (programa que utiliza Windows para acceder al registro), pero aprenderemos mucho más que con el publicado anteriormente.

Por primera vez trabajaremos con un desensamblador y un editor hexadecimal, y lo haremos sin grandes esfuerzos (prometido).

Este mes hemos impregnado varios artículos de TCP/IP, pero no hay un artículo completo dedicado al tema porque nuestro colaborador encargado del curso ha tenido que ausentarse por causas mayores. Tranquilos, el mes que viene estará de nuevo con nosotros.

Y no hay mal que por bien no venga, prepárate porque llega un nuevo curso MUY INTERESANTE. Te presentamos el SNORT. Si no sabes lo que es, ya puedes empezar a leer la revista :)

Y no me despido sin antes hacer un comentario que quizás te venga muy bien si estás pensando en comprar un monitor TFT. Hace unos días tuve que aguantar perplejo como un vendedor de hardware intentaba vender un monitor TFT de 20 pulgadas con una resolución de 1280*1024.

Un TFT de 20 pulgadas, entre otras cosas debe tener una resolución de 1600*1200. No exijas menos… lastima que en esta revista no tratamos temas de hardware… ¿a quien se le ocurre fabricar un 20 pulgadas a 1280*1024? Como mínimo es una tomadura de pelo.

Bueno, que me extiendo demasiado. Gracias de nuevo a todos los colaboradores, a AZIMUT que está poniendo un poco de orden en el CAOS y a los moderadores del foro que hacen todo lo posible por mantener un espacio donde reunirse y resolver dudas.

GRACIAS UNA VEZ MÁS.

INDICE

INDICE DE ANUNCIANTES

PROGRAMACION EN GNU LINUX
COLAS DE MENSAJES

el_chaman. Luis U. Rodriguez Paniagua

- Intercambio de Información entre programas: Ya hemos tratado los semáforos y la memoria compartida.
- Este mes le toca el turno a las "Colas de Mensajes".

1. Presentación

Las colas de mensajes, junto con los semáforos y la memoria compartida serán los recursos que ponen a nuestra disposición los sistemas UNiX para que los programas puedan intercambiar información entre sí.

2. Colas de Mensajes.

Una cola será una estructura de datos gestionada por el núcleo, donde van a poder escribir o leer varios procesos. Los mecanismos de sincronismo para que no se produzca colisión serán responsabilidad del núcleo.

Obsérvese que en el número pasado, al manejar memoria compartida, el propio programador tenía que gestionar la sincronización entre los procesos que accedían a la memoria compartida. En este caso, esto ya no es así: Será el sistema (núcleo del S.O.) el encargado de realizar esta tarea, pudiéndonos concentrar más en tareas de comunicación entre procesos que de concurrencia.

El manejo de colas de mensajes permite que dos procesos distintos sean capaces de enviarse mensajes (estructuras de datos que serán tratadas como un todo indivisible) y de esta forma pueden intercambiar información.

El mecanismo para conseguirlo es el de una cola de mensajes. Los procesos introducen mensajes en la cola y se van almacenando en ella. Cuando un proceso extrae un mensaje de la cola, extrae el primer mensaje que se introdujo y dicho mensaje se borra de la cola. A esta forma de comportarse una cola, y lo que la define, se le conoce como FIFO (*First Input, First Output*, Primero en entrar, Primero en Salir).

También es posible hacer "tipos" de mensajes distintos, de forma que cada tipo de mensaje contiene una información distinta y va identificado por un entero. Por ejemplo, los mensajes de tipo 1 pueden contener el saldo de una cuenta de banco y el número de dicha cuenta, los de tipo 2 pueden contener el nombre de una sucursal bancaria y su calle, etc.

Los procesos luego pueden retirar mensajes de la cola selectivamente por su tipo. Si un proceso sólo está interesado en saldos de cuentas, extraería únicamente mensajes de tipo 1, etc. Ni que decir tiene que si tenemos varios procesos que leen de una cola, el tipo de mensaje nos puede servir para que un determinado tipo de mensajes sea sólo atendido por un proceso concreto y el resto ignorado.

Tras resaltar esto último, una pequeña aclaración: Es posible hacer que un proceso que lea de la cola pueda especificar que desea leer mensajes independientemente de su tipo.

2.1. Solitud de una Cola de Mensajes

La primera operación que debemos aprender es la necesaria para solicitar una cola de mensajes. La manera de hacer esto es mediante la función:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int msgget(key_t key,  int shmflg);
```

Retorna: -1 en caso de error, o el identificador de la cola creada en otro caso.

El primer parámetro, key de tipo Clave, funciona tal y como se explicó en anteriores números, siendo su objetivo el de proporcionar al sistema un identificador común para los recursos IPC que están siendo utilizados por un grupo de procesos que acceden conjuntamente a un recurso IPC.

Una vez obtenida la clave, se crea la cola de mensajes. Para ello está la función mostrada msgget. Con dicha función creamos la cola y nos devuelve un identificador para la misma.

Si la cola correspondiente a la Clave key ya estuviera creada, simplemente nos daría el identificador de la misma (siempre y cuando los parámetros no indiquen lo contrario).

El segundo parámetro son unos *flags*. Aunque hay más posibilidades, lo imprescindible es:

- 9 bits menos significativos, son permisos de lectura/escritura/ejecución para propietario/grupo/otros, al igual que los ficheros. Para obtener una cola con todos los permisos para todo el mundo, debemos poner como parte de los flags el número 0777.

  Es importante el cero delante, para que el número se interprete en octal y queden los bits en su sitio (En C, cualquier número que empiece por cero, se considera octal). El de ejecución no tiene sentido y se ignora.

- IPC_CREAT. Junto con los bits anteriores, este bit indica si se debe crear la cola en caso de que no exista.

  Si está puesto, la cola se creará si no lo está ya y se devolverá el identificador. Si no está puesto, se intentará obtener el identificador y se obtendrá un error si no está ya creada.

En resumen, los flags deberían ser algo así como 0777 | IPC_CREAT

El identificador devuelto por msgget será heredado por los procesos descendientes del actual.

Para crear una cola de mensajes podemos emplear un código como el siguiente:

```
int msqid;
   key_t clave;
   ...
   clave = ftok ("archivo",'M');
   if (( msqid = msgget(clave,  IPC_CREAT | 0600 )) == -1)
   {
        fprintf(stderr, " ERROR al crear cola de mensajes\n");
   }
   ....
```

2.2. Control de las colas de mensajes.

Una vez que hemos creado una cola de mensajes, podemos realizar diversas operaciones de control sobre dicha cola con el fin de establecer distintos atributos de la misma u obtener diversa información estadística. Estas operaciones de control las realizará la función msgctl.

La declaración de esta función es:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

Retorna 0 en caso de éxito y -1 en caso de error.

msqid es el identificador de cola sobre el que vamos a operar.

cmd es la operación de control que queremos realizar. Este parámetro puede tomar los siguientes valores:

- IPC_STAT Lee el estado de la estructura de control de la cola y lo devuelve a través de la zona de memoria apuntada por *buf. La estructura de esta zona de memoria es:

```
/* Structure of record for one message inside the kernel.
The type `struct msg' is opaque.  */
   struct msqid_ds
   {
   struct ipc_perm msg_perm;   /* structure describing operation permission */
        __time_t msg_stime;          /* time of last msgsnd command */
        unsigned long int __unused1;
        __time_t msg_rtime;          /* time of last msgrcv command */
        unsigned long int __unused2;
        __time_t msg_ctime;          /* time of last change */
        unsigned long int __unused3;
        unsigned long int __msg_cbytes; /* current number of bytes on queue */
        msgqnum_t msg_qnum;            /* number of messages currently on queue */
        msglen_t msg_qbytes;          /* max number of bytes allowed on queue */
        __pid_t msg_lspid;          /* pid of last msgsnd() */
```

```
        __pid_t msg_lrpid;          /* pid of last msgrcv() */
        unsigned long int __unused4;
        unsigned long int __unused5;
    };
    struct ipc_perm {
       key_t  key;
       ushort uid;   /* owner euid and egid */
       ushort gid;
       ushort cuid;  /* creator euid and egid */
       ushort cgid;
       ushort mode;  /* lower 9 bits of access modes */
       ushort seq;   /* sequence number */
    };
```

- IPC_SET Inicializa algunos de los campos de la estructura de control de la cola. El valor de estos campos se toma de la estructura referenciada por *buf.
- IPC_RMID Elimina el sistema de cola de mensajes asociado al identificador msqid. Si la cola está siendo usada por varios procesos, no se elimina hasta que todos los procesos liberen la cola.

2.3. Operaciones con colas de mensajes

Para poder usar una cola de mensajes, las operaciones básicas que realizaremos serán escribir y leer mensajes. Para realizar estas operaciones disponemos respectivamente de las funciones msgsnd y msgrcv:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int msgsnd(int msqid, struct msgbuf *msgp, size_t msgsz, int msgflg);

ssize_t  msgrcv(int msqid, struct msgbuf *msgp, size_t msgsz, long msgtyp, int msgflg);
```

En ambas funciones msqid se corresponde al identificador de la cola sobre la que vamos a operar.

msgp es un puntero que señala la zona de memoria donde están o donde queremos recibir los datos que se enviarán o leerán de la cola. La composición de esta zona de memoria es definida por el usuario mediante una estructura de datos. La única restricción que se impone es que el primer campo sea de tipo long, siendo su funcionalidad la de identificar el tipo de mensaje.

msgsz será el tamaño en *bytes* del mensaje que queramos enviar o recibid. Ojo: En este tamaño **no se incluyen los bits que ocupa el campo** long antes descrito (el tipo de mensaje).

msgtyp sólo aparece en la llamada de lectura y especifica el tipo de mensaje que queremos leer. Puede tomar los siguientes valores:

- msgtyp = 0 Leer el primer mensaje que haya en la cola.
- msgtyp > 0 Leer el primer mensaje de tipo msgtyp que haya en la cola.
- msgtyp < 0 Leer el primer mensaje cumpla que su tipo es menor o igual al valor absoluto de msgtyp y a la vez sea el más pequeño de los que hay.

msgflg es un mapa de bits que tiene distinto significado dependiendo de que estemos utilizando la función msgsnd o msgrcv:

- Para el caso de escritura en cola (msgsnd), si la cola está llena:
  - IPC_NOWAIT: Si activamos este bit, la llamada de a msgsnd devolverá el control inmediatamente retornando -1.
  - IPC_NOWAIT: Si no activamos este bit, el proceso suspende su ejecución hasta que haya espacio libre disponible en la cola..
- Para el caso de lectura desde la cola (msgrcv), si la cola está vacía:
  - IPC_NOWAIT: Si activamos este bit, la llamada de a msgsnd devolverá el control inmediatamente retornando -1.
  - IPC_NOWAIT: Si no activamos este bit, el proceso suspende su ejecución hasta que haya un mensaje del tipo disponible en la cola.

En el caso de que todo funcione correctamente, *msgsnd* devuelve 0 y *msgrcv* el número de *bytes* recibidos sin incluir los bytes del tipo de mensaje.

Veamos cuales serían las líneas de código necesarias para enviar y recibir un mensaje de tipo 1 compuesto de 20 caracteres:

```
int msqid;
struct
{
  long tipo;
  char cadena[20];
}mensaje;

/* Debemos de quitar el tamaño del tipo */
int longitud = sizeof(mensaje) - sizeof(mensaje.tipo);

....

/* Envío del mensaje */
mensaje.tipo=-1;
strcpy(mensaje.cadena,2HOLA");
if(msgsnd(msqid, &mensaje, longitud, 0)==-1)
{
      /* error */
}
....
/* Recepción del mensaje */
if(msgrcv(msqid, &mensaje, longitud,1, 0)==-1)
{
      /* error */
}
```

2.4. Ejemplo de uso

El presente ejemplo está basado en uno similar sacado del libro *"UNiX, programación avanzada, Ed. Ra-Ma"* de Fco. Manuel Márquez. La razón de ello, como pronto se verá, es que es una aplicación bastante completa y a la vez sencilla, utilizando al máximo las capacidades del tema expuesto hoy.

Este ejemplo está disponible de manera gratuita para el público en la página de la editorial Ra-Ma. Me he permitido modificar partes del código y añadir comentarios para que dicho ejemplo sea coherente con los visto en el presente curso, manteniendo en cualquier caso el interés didáctico que el autor original supo plasmar en él.

El ejemplo pretende ilustrar la construcción de un DBM (*Data Base Manager*) de manera que pueda gestionar los distintas tareas que le mandan hacer distintos usuarios (programas clientes) de una minimalista base de datos. Dicho gestor se comunicará directamente con la base de datos. En nuestro caso será un simple archivo binario que deberemos especificar por línea de comandos cuando arranquemos el gestor y sobre el que escribirá y leerá directamente.

Para comunicarse con el gestor se utilizarán dos colas: Una llamada cliente-gestor será la empleada para que los clientes envíen mensajes al gestor, y otra llamada gestor-clientes para que el gestor se comunique con los clientes.

Este funcionamiento se muestra en el siguiente esquema:

Archivo de definición de tipos y constantes (datos.h):

```
/*
 * FICHERO: datos.h
 *
 * DESCRIPCI~N:
 *  Fichero de cabecera com~n para los clientes y DBM de la base de datos.
 *
 * * * * * * * * ** * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#ifndef _DATOS_H_
#define _DATOS_H_

/* Creamos un tipo de datos para almacenar los datos
 * de una persona.
 */
typedef struct
{
      char nombre[61];
      char direccion[121];
      char telefono[11];
}persona ;

/* Creamos un tipo de datos para almacenar los datos
 * de un mensaje.
 */
typedef struct
{
      long pid;
      int  orden;
      union
      {
            persona una_persona;
      }datos;
}mensaje;


#define LONGITUD (sizeof(mensaje) -sizeof(long))

/* Comandos para el DBM */
#define LISTAR 1
#define ANADIR 2
#define FIN    3
#define ERROR  4

#define FICHERO_CLAVE "datos.h"
```

```
#define CLAVE_CLIENTE_GESTOR 'K'
#define CLAVE_GESTOR_CLIENTE 'L'

#endif
```

Gestor (gestor.c):

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "datos.h"

int main(int argc,char *argv[])
{
      int    cola_cg, cola_gc, funcionando=1;
      mensaje un_mensaje;
      key_t  clave;
      FILE   *base_de_datos;

      /* Como este programa recibe datos desde la
       * l~nea de comandos, tenemos que verificarlos
       */
      if(argc != 2)
      {
            fprintf(stderr,"Forma de uso: %s fichero.\n",argv[0]);
            return -1;
      }


      /* Creaci~n de las colas de mensajes (ver documentaci~n
       * adjunta en el art~culo)
       */
      clave=ftok(FICHERO_CLAVE, CLAVE_CLIENTE_GESTOR);
      if((cola_cg=msgget(clave,IPC_CREAT|0666))==-1)
      {
            fprintf(stderr,"Error al crear la cola CLIENTE-SERVIDOR");
            return -1;
      }
      clave=ftok(FICHERO_CLAVE, CLAVE_GESTOR_CLIENTE);
      if((cola_gc=msgget(clave,IPC_CREAT|0666))==-1)
      {
            fprintf(stderr,"Error al crear la cola SERVIDOR-CLIENTE");
            return -1;
      }


      /*
       * Comienza el servicio del GESTOR. Nos limitamos a mirar si hay
       * cosas que leer de la cola CLIENTE-SERVIDOR y si es as~
       * actuaremos en consecuencia.
       */
      while(funcionando) /* Este proceso dura indefinidamente */
      {
            /* Obs~rvese que no activamos el bit el bit IPC_NOWAIT
             * Esto quiere decir que si no hay nada que leer de la cola
             * esperamos a que lo haya.
             */
            msgrcv(cola_cg, &un_mensaje, LONGITUD, 0, 0);

            /* Una vez le~do un mensaje, tendremos que atender a lo que
             * nos manda hacer*/

            /* Listar la base de datos que estar~ almacenada en el fichero
             * que hemos pasado por l~nea de comandos */
            if(un_mensaje.orden == LISTAR)
            {
                  /* Si el fichero de base de datos no se puede leer...*/
                  if((base_de_datos=fopen(argv[1],"r"))==NULL)
                  {
                        /* Imprimimos en la salida est~ndar de error
                         * que ha ocurrido esto
                         */
                        fprintf(stderr,"GESTOR listar: error al abrir el fichero de bases de datos");
                        /* Informamos al cliente de lo que ha sucedido */
                        un_mensaje.orden=ERROR;
                        msgsnd(cola_gc, &un_mensaje, LONGITUD, 0);
                  }
                  else
                  {
                        /* Realizamos un listado, enviando al cliente todos
                         * los registros de la base de datos
                         */
                        while(fread(&un_mensaje.datos.una_persona, sizeof(persona), 1, base_de_datos)==1)
                        {
                          msgsnd(cola_gc, &un_mensaje, LONGITUD, 0);
                        }
                        fclose(base_de_datos);
                        /*Preparamos un mensaje especial  para envi~rselo
                         * al cliente con el fin de indicarle que se ha
                         terminado el listado. */
                        un_mensaje.orden=FIN;
                        msgsnd(cola_gc, &un_mensaje, LONGITUD, 0);
                  }
            }
            else
            {
                  if(un_mensaje.orden==ANADIR)
                  {
                        if( ( base_de_datos=fopen(argv[1],"a") )==NULL ){
                        fprintf(stderr,"GESTOR a~adir: error al abrir el fichero de bases de datos\n");
                              un_mensaje.orden=ERROR;
                              msgsnd(cola_gc, &un_mensaje, LONGITUD, 0);
                        }
                        else
                        {
                              fwrite(&un_mensaje.datos.una_persona, sizeof(persona),1,base_de_datos);
                              fclose(base_de_datos);
                              un_mensaje.orden=FIN;
                              msgsnd(cola_gc,&un_mensaje,LONGITUD,0);
                        }
                  }
                  else
                  {
                        if(un_mensaje.orden==FIN)
                        {
```

```
                    /* Tenemos que apagar todo esto */
                    msgctl(cola_gc,IPC_RMID,NULL);
                    msgctl(cola_cg,IPC_RMID,NULL);
                    funcionando=0;
                    printf("\n Programa Gestor parado a petici~n del cliente\n");
                }
                else
                {
                    /* ~una opci~n desconocida? */
                    un_mensaje.orden=ERROR;
                    msgsnd(cola_gc,&un_mensaje, LONGITUD,0);
                }
            }
        }
    }
    return 0;
}
```

Cliente (cliente.c):

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include "datos.h"

int main(int argc, char *argv[])
{
    int cola_gc, cola_cg, pid;
    char opcion, minibuf[2];
    mensaje un_mensaje;
    key_t clave;
    enum {NO, SI} recibir = NO;


    /* Creaci~n de las colas de mensajes */
    clave = ftok(FICHERO_CLAVE, CLAVE_CLIENTE_GESTOR);
    if((cola_cg=msgget(clave,0666))==-1)
    {
        fprintf(stderr,"CLIENTE %i: error creando cola_cg",getpid());
        return -1;
    }
    clave = ftok(FICHERO_CLAVE, CLAVE_GESTOR_CLIENTE);
    if((cola_gc=msgget(clave,0666))==-1)
    {
        fprintf(stderr,"CLIENTE %i: error creando cola_gc",getpid());
        return -1;
    }

    un_mensaje.pid = pid = getpid();
    while(1)
    {
        printf("Ordenes:\n");
        printf("l - Listar\n");
        printf("a - A~adir\n");
        printf("f - Fin\n");
        printf("s -Salir\n");
```

```
printf("\tOrden: ");
scanf("%s",minibuf);
opcion=minibuf[0];

if(opcion=='l')
{
    /* Esta orden producir~ una respuesta por parte
     * del servidor. Nos preparamos para recibir datos.
     */
    recibir=SI;
    un_mensaje.orden = LISTAR;
    msgsnd(cola_cg,&un_mensaje, LONGITUD, 0);
}
else
{
    if(opcion=='a')
    {
        recibir=SI;
        printf("\tNombre: ");
        scanf("%s",un_mensaje.datos.una_persona.nombre);
        printf("\tDirecci~n: ");
        scanf("%s",un_mensaje.datos.una_persona.direccion);
        printf("\tTel~fono: ");
        scanf("%s",un_mensaje.datos.una_persona.telefono);
        un_mensaje.orden=ANADIR;
        msgsnd(cola_cg,&un_mensaje,LONGITUD,0);
    }
    else
    {
        if(opcion=='f')
        {
            recibir=NO;
            un_mensaje.orden=FIN;
            msgsnd(cola_cg, &un_mensaje, LONGITUD, 0);
        }
        else
        {
            if(opcion=='s')
            {
                return 0;
            }
            else
            {
                recibir = NO;
                printf("Opci~n [%c] err~nea\n", opcion);
            }
        }
    }
}

/*
 * Si nos hemos preparado a recibir algo, tenemos que
 * mirar si en la cola hay algo para nuestro cliente
 */
if(recibir==SI)
{
    do
```

```
{
    /*
     * F~jense en que ahora s~lo queremos
     * leer los mensajes del tipo de nuestro
     * cliente, por ello pasamomos como par~metro el PID
     */
    msgrcv(cola_gc, &un_mensaje, LONGITUD, pid, 0);
    if(un_mensaje.orden==LISTAR)
    {
        /* Hay que imprimir esto */
    printf("\n Registro: %s %s %s \n", un_mensaje.datos.una_persona.nombre,
    un_mensaje.datos.una_persona.direccion,
    un_mensaje.datos.una_persona.telefono);
    }
    else
    {
        if(un_mensaje.orden==ERROR)
                {
                    printf("\n Mensaje de error recibido desde el servidor\n");
                }else
                {
                    if(un_mensaje.orden==FIN)
                    {
                        printf("----\n");
                    }
                    else
                    {
                        printf("Se recibi~ un mensaje desconocido\n");
                    }
                }
            }
        }while((un_mensaje.orden!=FIN) && (un_mensaje.orden!=ERROR));
    }
}
}
```

XBOX LIFE V
JUGANDO ONLINE GRATIS
POR ALFONSO MENKEL

- **¿Quieres jugar GRATIS por Internet con la XBOX?**
- **Descubre el XBCONNECT**
- **Crea Partidas por Internet e Invita a tus Amigos.**

Saludos compañeros, aquí estamos un mes más. Este mes os voy a enseñar a jugar online GRATIS, pero quiero que quede claro que no vamos a piratear el servicio XBOX LIVE, sino que vamos a usar un programa alternativo. No es tan bueno como el Live, pero está muy bien.

Lo que necesitamos:

- Consola Xbox -El mod chip es opcional, pero es necesario si queréis jugar con copias de seguridad ;)
- PC con tarjeta de red o Router.
- Conexión de banda ancha (ya sea ADSL o Cable).
- XBCONNECT ultima versión.
- Un juego compatible con este método.

Como ya he explicado en los números anteriores, hace falta conectividad de Xbox a PC, sin esto no será posible jugar online.

Nos vamos a la Web oficial de Xbconnect (http://www.xbconnect.com) y nos descargamos la última versión, lo siento por los usuarios de linux, pero de momento no esta el programa para el SO del Pingüino.

El programa también encontrarás en la Web de la revista (www.hackxcrack.com), en la sección **Artículos liberados y Descargas**



Hay dos versiones del ...

Hay dos versiones del programa, el Normal (gratis) y el PRO (de pago), la verdad es que no sé qué diferencia hay si pagas por el pro, pero la versión normal va muy bien.

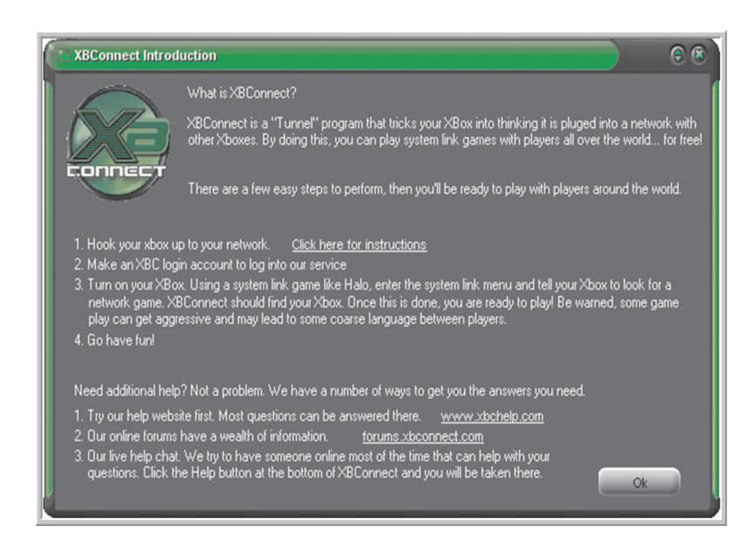
Mientras está bajando el programa, nos fijamos en el lado derecho de la Web y veremos los juegos compatibles con este método.

Como vemos en la lista, todos los juegos tienen algo en común: todos tienen la posibilidad de jugar en red local. Pues gracias a eso y al Xbconnect podemos hacer eso mismo, pero a través de Internet.

Ahora que ya sabemos cómo funciona el sistema, vamos a instalar el programa:

La instalación es muy sencilla, damos a Next hasta que se acabe la instalación :)

Arrancamos el programa y veremos la primera pantalla.



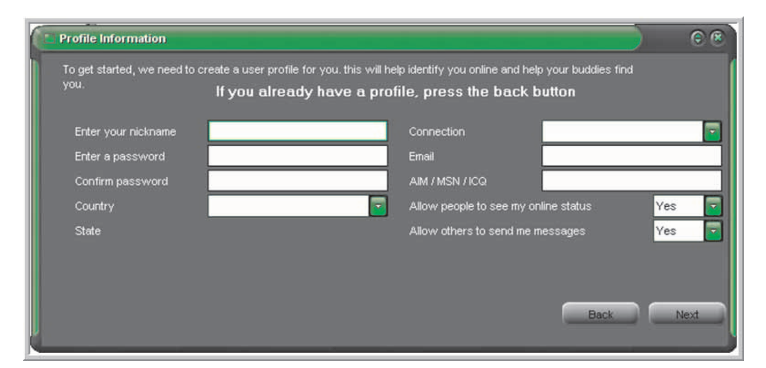
Pues nada, pinchamos en el OK.

Acto seguido veremos la siguiente pantalla donde se nos pregunta en perfecto Ingles si deseamos crear una nueva cuenta.



Si no tenemos una cuenta creada anteriormente, pulsaremos el botón YES.

Acto seguido nos saldrá un formulario que deberemos cumplimentar.

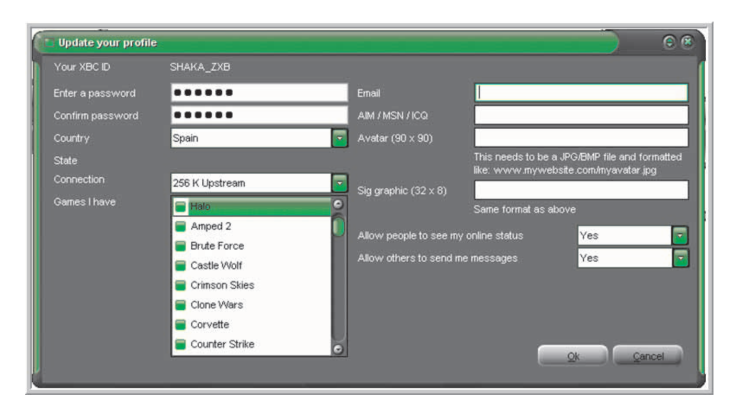


Una vez rellenamos los datos que nos pide pulsaremos el botón NEXT. Nos aparecerán un par de ventanitas de confirmación y finalmente llegaremos a la siguiente imagen.

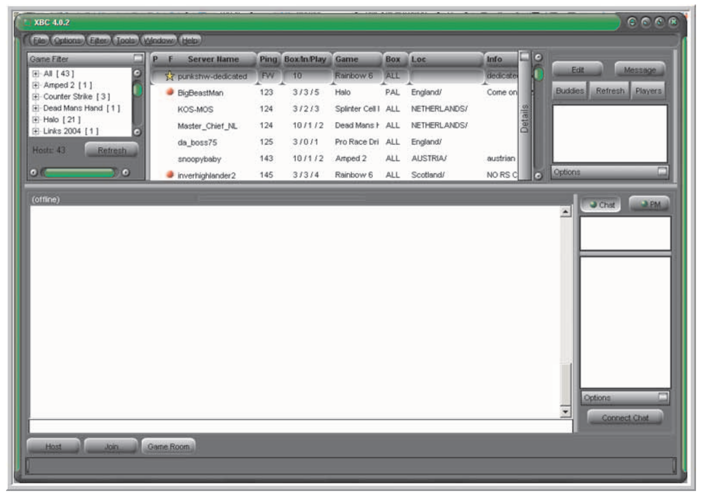


Si hemos sido originales, no habrá ningún problema con el alta de usuario, pero si el Nick estuviese ocupado ya, deberemos elegir otro hasta que lo acepte.

Introducimos el Nick y Pass que acabamos de crear y pinchamos en Login. El programa nos pide que nos tomemos unos segundos para actualizar nuestro perfil... ... llegando a la siguiente pantalla.



Lo único que debemos hacer ahora es seleccionar los juegos que tenemos, aceptamos y llegaremos a la pantalla principal del programa.



Ahora debemos conectar la consola al PC mediante el cable RJ45



Si la conexión...

Si la conexión es directa de la consola al PC, EL CABLE RJ-45 debe ser cruzado. Si entre el PC y la Xbox hay un swich, Hub o Router, el cable debe ser no cruzado.
Si no tienes ni idea de cables y esto te suena muy raro, no tengas vergüenza. Te aseguramos que si vas a una tienda y pides un "Cable de Red RJ-45 no Cruzado" o un "Cable de Red RJ-45 Cruzado", te entenderán perfectamente.
No siempre fue así, hace algunos años (bastantes), si pedías "eso" te miraban mal :)

Con esto claro, debemos pinchar en el Menu Options --> General Options.



Ahora en la consola ponemos el juego al que queramos jugar. En el juego debemos ir a INTERCONEXION o JUGAR EN RED y BUSCAR PARTIDA.

Ahora en el PC pinchamos el botón FIND (puedes verlo en la imagen anterior) y, si todo está configurado bien, nos encontrará la tarjeta de red del PC, y la consola Xbox.

Fijaos que arriba del todo, en el lado derecho, pone "**My Xbox Status FOUND**", hasta que este estatus no esté en **Found**, no se podrá jugar online.



Los problemas...

Los problemas más frecuentes a la hora de encontrar la consola son:

- *Que tengas un cortafuegos y esto no permita el buen funcionamiento. Para solucionar esto, configuramos el cortafuegos o lo desactivamos.*
- *Que el cable no este bien o no sea cruzado cuando debe serlo, o viceversa.*
- *Que no tengáis la conexión de Internet compartida.*

Ahora pinchamos en OK.

Este programa tiene opción de chat, para así organizar mejor las partidas.

Si queremos podemos conectarnos al Chat, basta con pulsar el botón de la esquina inferior derecha (**Connect Chat**).

Podemos, o bien jugar una partida creada, o bien crear una partida y esperar a que se junte la peña. Veamos las dos formas.

JUGANDO UNA PARTIDA CREADA:

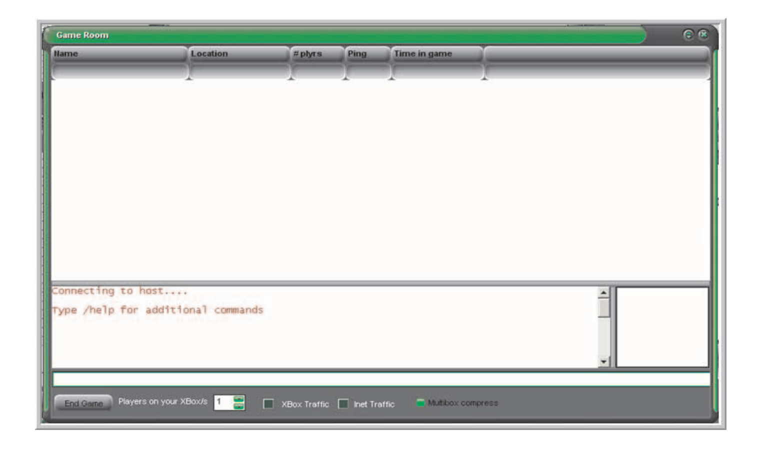
En la parte superior vemos todas las partidas que hay en juego, seleccionamos una que nos guste, pinchamos dos veces encima de la partida que queremos jugar.



Los juegos ...

Los juegos vienen en varios formatos dependiendo de su procedencia. El formato PAL es la europea, NTSC Japonesa y Americana.

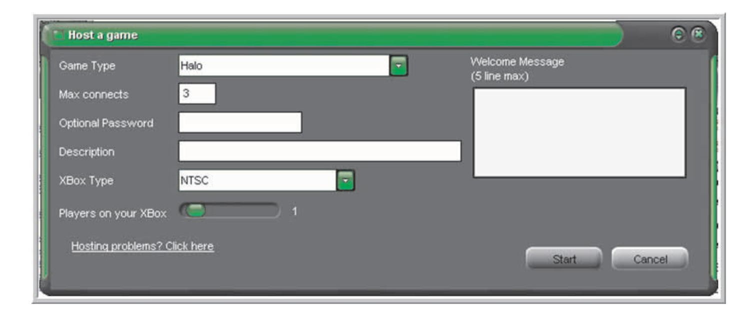
Para poder jugar una partida online, debemos tener el mismo formato que el juego creado.



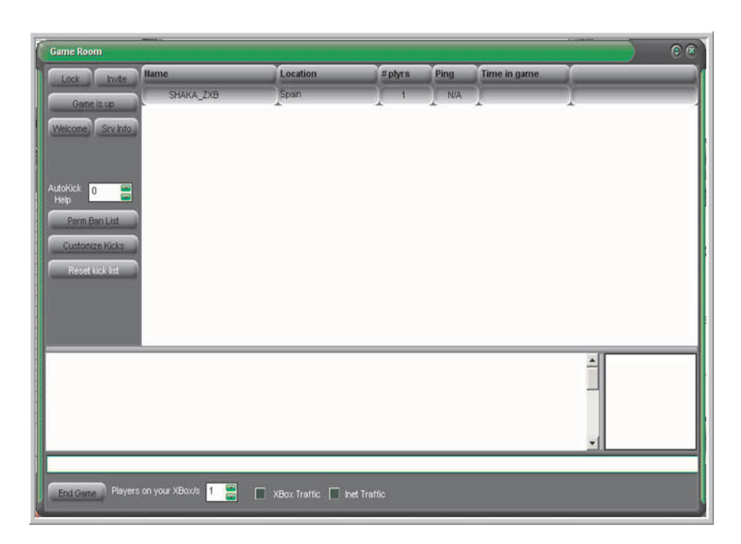
El juego debe estar buscando partida, cuando el tipo que haya creado la partida la inicie, vais a la consola y a jugar.

CREAR PARTIDAS:

Pinchamos en HOST



Seleccionamos el juego, la cantidad de jugadores que permitimos, la contraseña es opcional, la descripción del formato (NTSC o PAL), la cantidad de jugadores que van a jugar en tu Xbox y el mensaje de bienvenida.



Esperamos a que entre la gente a jugar y cuando nos guste, nos vamos a la consola e iniciamos la partida.

Hay algunas opciones que no he explicado porque son muy sencillas y os enteraréis fácilmente de cómo son estas cosas.

Hay otro programa que se llama **Xlink** que también es muy bueno y se configura de la misma forma.

También podéis encontrar estos programas para otras consolas, como **Game Cube** y **PS2**.

Espero veros jugando online y poder retaros a unas partidas, nos lo pasaremos muy bien.

El mes que viene meteremos varios juegos en un mismo DVD con un menú personalizado por nosotros mismos.

Hasta el mes que viene.

Salu2.

PD. Este mes quiero agradecerle a mi novia Cristina su gran labor a la hora de corregirme los artículos, MUCHAS GRACIAS ERES LA MEJOR :)

CRACKEANDO LAS RESTRICCIONES DE SEGURIDAD DE WINDOWS

por Hexborg

Estáis en el ordenador del trabajo. O en el de clase. O en un cyber. Os apetece hackear. Queréis instalar un programa que os gusta o que necesitáis. Os ponéis manos a la obra y entonces... ¡Horror! ¡Los programas no se instalan! ¡No podéis ejecutar el editor del registro! ¡No podéis lanzar el panel de control! ¡El menú de contexto no funciona! ¡Esto es un infierno! El administrador, para que no toquetees el sistema, ha deshabilitado la mayoría de opciones!!! ¿Y ahora qué hacemos? Pues leer este artículo :)

0. ADVERTENCIA!!!

No empieces a hacer este ejercicio "a lo loco". Antes léelo completo y decide si vas a hacerlo o no. Ni la editorial ni el autor de este artículo se hace responsable del mal uso de las explicaciones expuestas ni de los perjuicios que puedas causar en tu ordenador (o en el de terceros).

Venga, resumiendo, que las pruebas en casa y con gaseosa. A estas alturas ya deberías tener un Sistema Operativo "de pruebas" para hacer los ejercicios que te proponemos :)

1. ¿Qué son las restricciones de seguridad?

Windows permite controlar la ejecución de algunos de sus programas y componentes colocando ciertos valores en el famoso "Registro de Windows".

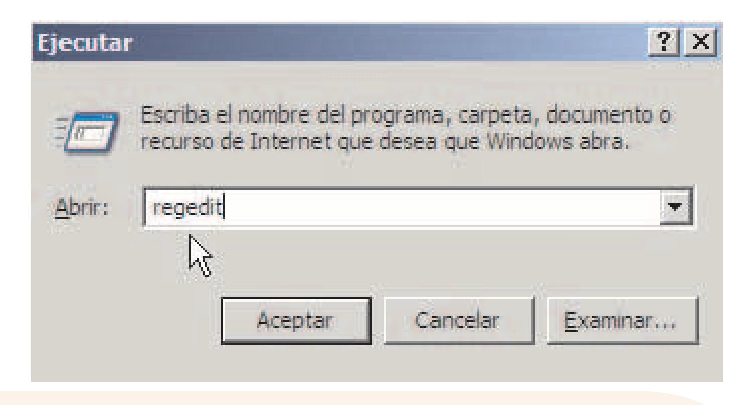
Cuando ejecutamos un programa, este comprueba primero los valores del "Registro de Windows". Si en el registro existe "tal valor", el programa se negará a ejecutarse y, si existe "tal otro", se ejecutará sin novedad.

En esto consisten las restricciones de seguridad. Se trata de un sistema poco efectivo que resulta muy sencillo de saltar.

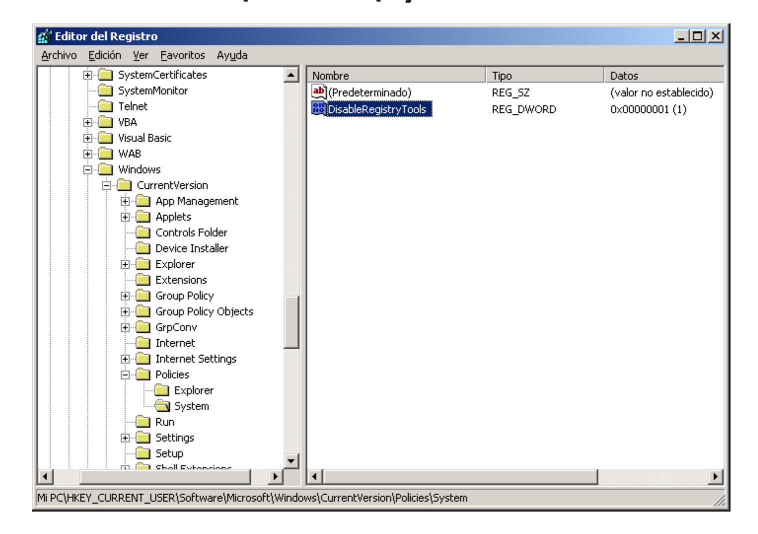
El truco está en que **es el propio programa el que comprueba si esos valores existen en el registro**. **Aquí está el punto débil: si hacemos que el programa no compruebe nada, la restricción será inútil :)**

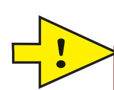
Vamos a verlo con un ejemplo. Vamos a hacer que el programa **regedit** (un componente de Windows que sirve precisamente para modificar el Registro de Windows) no pueda ser ejecutado y de esa forma "nadie" pueda cambiar nuestro registro de Windows. Esto lo que hacen algunos administradores para que no puedas "tocarles el Windows". **---> Y después "eludiremos" esta protección <---**

Abrid el regedit. Ya sabéis, no está en el "**menú de programas**", así que usad --> Botón Inicio --> Ejecutar (escribimos "**regedit**" y pulsamos aceptar).



Ahora buscad la clave **HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System**.

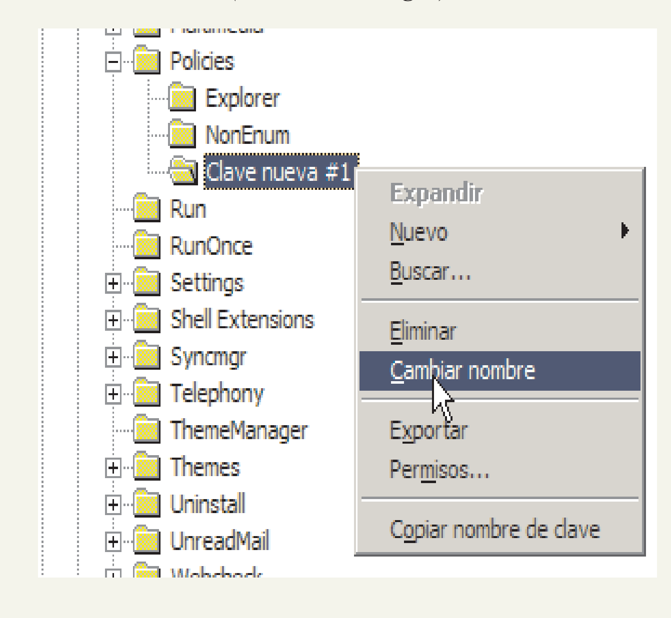




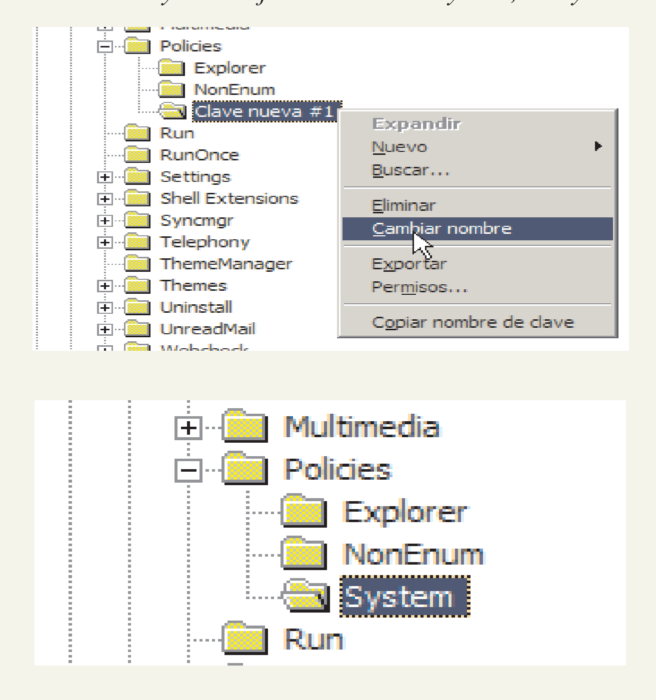
Puede ser que...

Puede ser que en tu Windows solo llegues hasta HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies y no encuentres la clave System (porque simplemente no existe).

Ningún problema, sitúa el Mouse sobre Policies, pulsa el botón derecho, aparecerán varias opciones, selecciona Nuevo ---> Clave (como en la imagen)



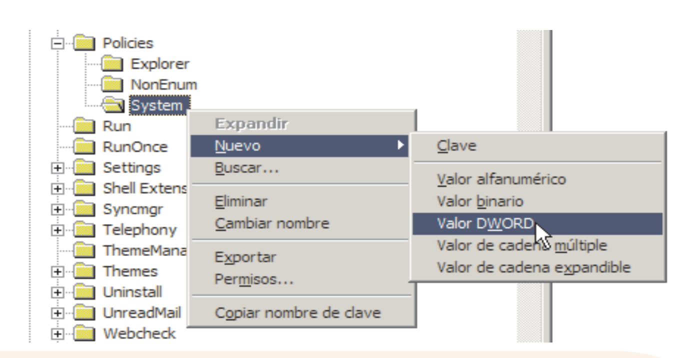
En cuanto selecciones la opción Clave (pulsando con el botón izquierdo del Mouse), verás una nueva Clave (con un nombre tipo "Clave nueva #1").
Pulsando sobre esta nueva Clave (con el botón derecho del Mouse) selecciona la opción Cambiar Nombre y dale el nombre System. Ojo!!! He dicho System, no system.



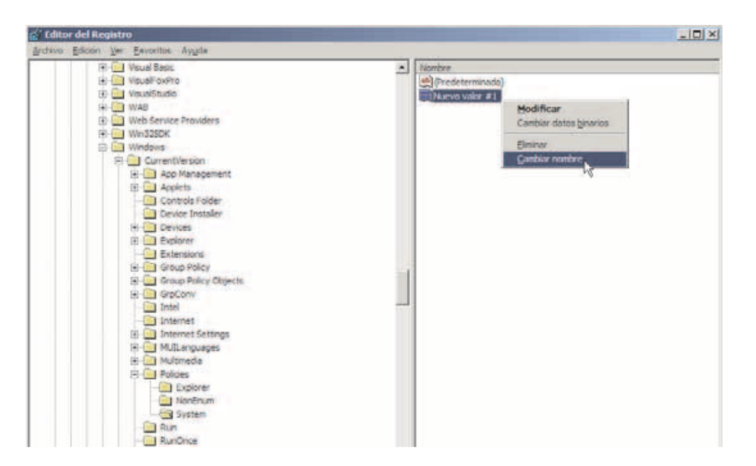
Muy bien, pues ya tenemos nuestra Clave System :)

En esa clave, crearemos un nuevo valor de tipo ***dword***, que se llame **DisableRegistryTools** y le daremos valor **1**. Venga, que no es complicado:

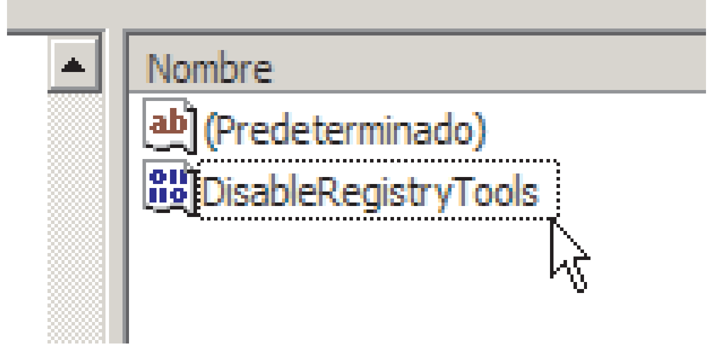
- Coloca el Mouse sobre la **Clave System** y pulsa el botón derecho. Selecciona (pulsa con el botón izquierdo del Mouse) la opción **Nuevo Valor --> Valor DWORD**



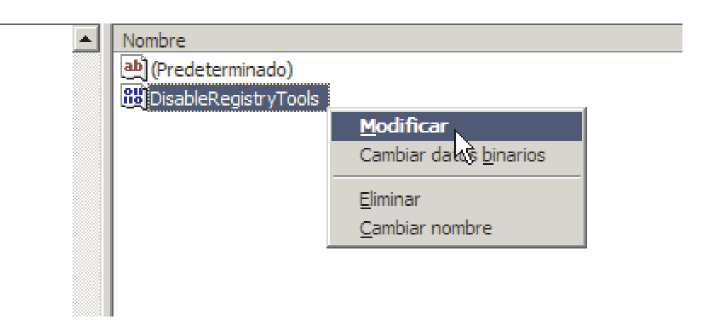
En la ventana de la derecha podrás ver que se ha creado un nuevo valor llamado (por defecto) **Nuevo valor #1**
Cámbiale el nombre por DisableRegistryTools. Esto se consigue pulsando con el botón derecho del Mouse sobre **Nuevo valor #1** y seleccionando la opción **Cambiar Nombre**. Acto seguido escribes el nombre que queremos darle, ya sabes, **DisableRegistryTools.**



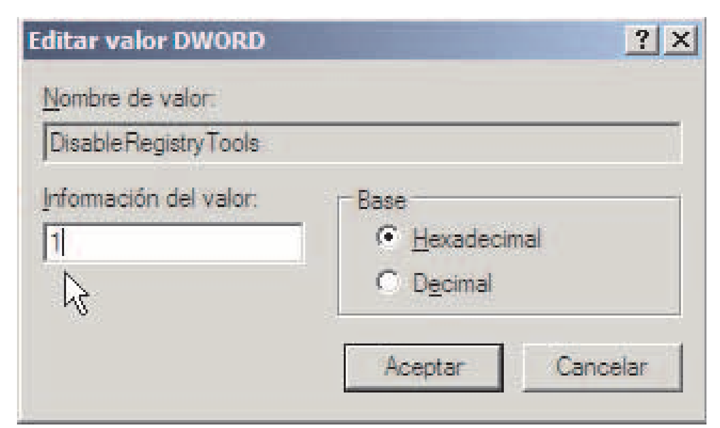
Esto es lo que nos quedará:



Por defecto, **DisableRegistryTools** tiene el valor 0 (es decir, inactivo). Vamos a darle el valor 1 (es decir, activado). En la ventana derecha, pulsamos el botón derecho del Mouse sobre **DisableRegistryTools** y seleccionamos la opción **Modificar**



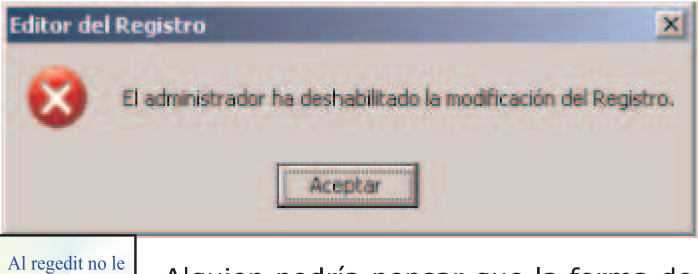
Pues venga, en la nueva pantalla que aparecerá cambiaremos el **0** por un **1**, pulsaremos **aceptar y listo!!!**



¿Ahora que? ¿Qué hemos hecho?

Ya puedes cerrar el **regedit**. Con esto lo que hemos hecho es **impedir que el usuario actual**, o sea, nosotros, podamos ejecutar una herramienta de edición del registro. Usease, el **regedit**. Si le diéramos a **DisableRegistryTools** el valor 0 o lo borrásemos del registro, podríamos volver a ejecutar el **regedit** sin problemas.

Vamos a probarlo. Ejecutad el **regedit** como antes y fijaos qué mensaje más chulo nos sale diciendo que el administrador ha deshabilitado la modificación del registro. En esto consisten las restricciones de seguridad de Windows basadas en el registro.



Al regedit no le caigo bien.

Alguien podría pensar que la forma de solucionar esto es muy simple: sólo hay que borrar las claves del registro que nos están molestando y se acabó el problema. Pero, ¿qué pasa si estamos ante un Windows de la familia de NT, en el que las claves del registro tienen permisos, y no tenemos permiso para modificarlas? Pues que habrá que pasar a palabras mayores ;)

2. ¿Qué hacemos ahora?

Bueno, ya podemos volver a ejecutar el **regedit** para borrar la clave que hemos creado. Un momento... **No podemos ejecutar el regedit porque la clave no nos lo permite!!!, y no podemos borrar la clave sin ejecutar el regedit. ¡Houston, tenemos un problema!** Pero tranquilos, que no cunda el pánico. Vamos a solucionarlo.

Antes que nada quiero advertiros de que el **regedit** con el que voy a trabajar es la versión de Windows XP. Si usáis otra versión, podéis encontrar diferencias, pero todo lo que digo aquí os sirve.

Lo primero que hay que hacer en estos casos es tomar nota del mensaje que nos aparece al ejecutar el programa. Lo recordáis, ¿verdad? : "**El administrador ha deshabilitado la modificación del Registro**". Ese mensaje es el que nos permitirá encontrar la instrucción que hay que modificar para conseguir nuestro objetivo, tranquilo, que ahora lo explicamos :)

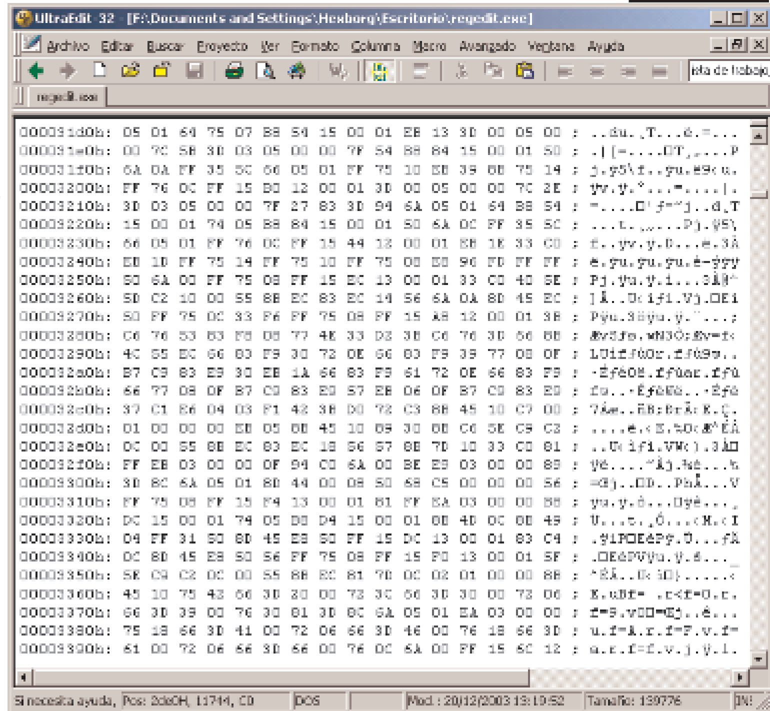
¿Cómo vamos a hacer esto? Pues no es tan complicado como parece. Un programa, como ya debéis saber, no es más que una secuencia de instrucciones que se ejecutan en serie para llegar a realizar la tarea que se le ha encomendado.

Cuando un programador escribe un programa, lo hace en un lenguaje de **alto nivel** como C, PASCAL, BASIC o cualquier otro de los muchos que existen. Estos lenguajes se llaman "de alto nivel" porque están próximos a la forma de pensar de las personas. Es decir, tienen instrucciones escritas en un lenguaje "similar" al inglés que se pueden entender fácilmente al leerlas.

También existen los lenguajes de **bajo nivel**, que son lenguajes cercanos al funcionamiento interno del ordenador. Están formados por instrucciones muy básicas que resultan muy sencillas de ejecutar.

Uno de estos lenguajes es el que se llama "**lenguaje máquina**" o "**código máquina**". Este lenguaje es el único que el microprocesador entiende directamente. Todos los demás deben ser traducidos a lenguaje máquina para poder ser ejecutados.

Esto es código máquina. Libera tu mente.



Recapitulando, cuando un programador escribe un programa, lo escribe en un lenguaje de alto nivel. Luego lo traduce a lenguaje máquina usando un programa llamado **compilador** o **ensamblador** dependiendo de que el lenguaje sea de alto o de bajo nivel y, tras un proceso que no me corresponde explicar aquí (ya ha sido explicado en anteriores números de esta revista), acaba obteniendo un ejecutable (por ejemplo el típico archivo **loquesea.exe**).

De manera que un ejecutable no es ni más ni menos que una serie de instrucciones traducidas a lenguaje máquina. Si conocemos el lenguaje máquina y podemos leer y modificar estas instrucciones, entonces seremos capaces de entender lo que hace el programa y cambiarlo a nuestro gusto.

Esto que parece muy sencillo en realidad puede ser terriblemente complicado. El problema es que el lenguaje máquina está formado por números. Esos números se agrupan en varios grupos y cada grupo de números es una instrucción con sus parámetros y todo.

Si miráis un programa en lenguaje máquina, cosa que se puede hacer abriendo un ejecutable con un editor hexadecimal, lo único que veréis será un montón de números en hexadecimal. A ver quién es el guapo que entiende algo en ese galimatías que parece el código de matrix. Pues hay gente que es capaz de hacerlo, os lo aseguro, pero nosotros no lo vamos a necesitar (menos mal).

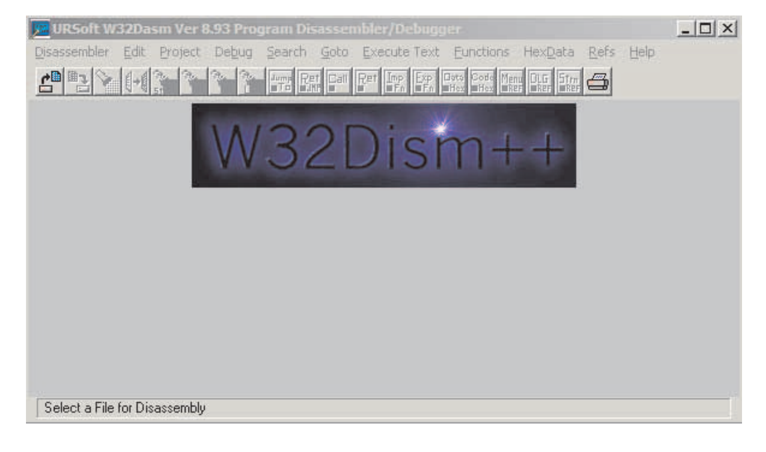
La solución es usar otro lenguaje llamado "**lenguaje ensamblador**". Este es otro lenguaje de bajo nivel que, a diferencia del lenguaje máquina, está formado por palabras que se pueden entender. La gracia está en que existe una equivalencia entre las instrucciones del ensamblador y las del lenguaje máquina, es decir, se puede traducir directamente de lenguaje máquina a ensamblador y viceversa.

Esta característica muy especial y no la tienen otros lenguajes, en los que es muy fácil traducir del lenguaje que estáis usando a lenguaje máquina, pero es muy difícil hacer el proceso inverso.

De manera que lo que necesitamos es un programa que traduzca de lenguaje máquina a lenguaje ensamblador y nos permita ver así el ejecutable. Este programa se llama **desensamblador**, y nosotros utilizaremos el **W32Dasm**, que es uno de los mejores desensambladores que existen.

3. Manos a la obra

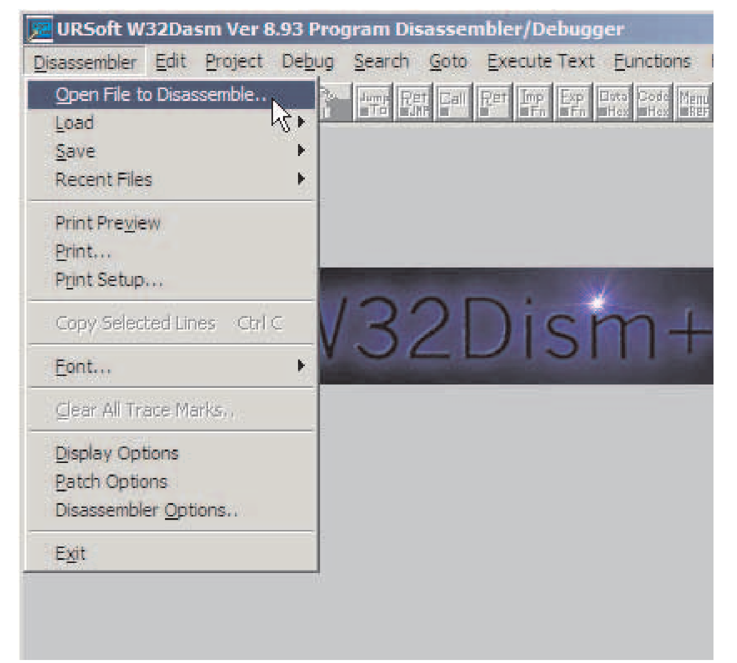
Antes que nada, pasaremos por www.hackxcrack.com, iremos a la sección "Artículos Liberados y Descargas", entraremos en el número 19 y nos descargaremos el programa W32Dasm. Nada mas descargarlo ya puedes ejecutarlo -no necesita instalarse :)-.



Ahora creamos una carpeta en **C:** llamada, por ejemplo, **loveasm**. Buscamos en nuestro PC el archivo

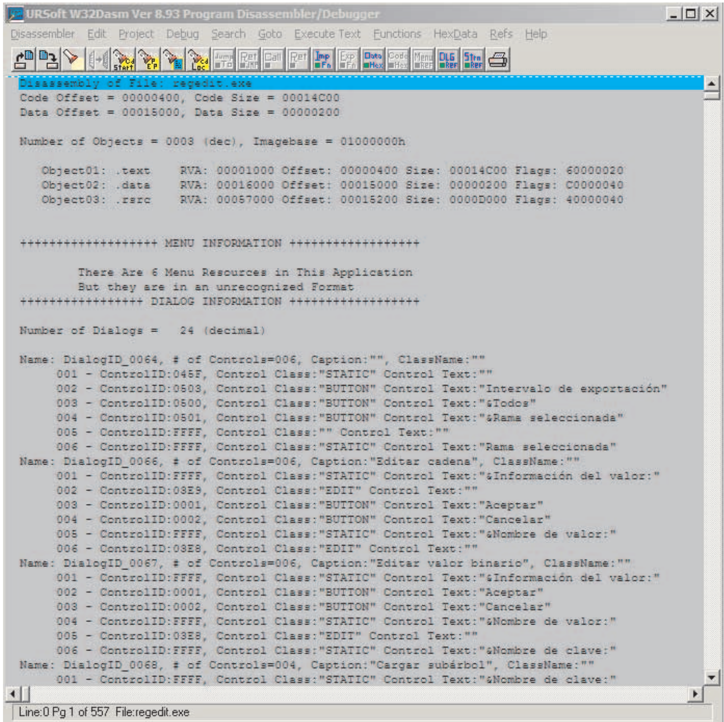
regedit.exe (normalmente estará en **C:\Windows**) y lo copiamos (he dicho copiar, no mover) en **C:\loveasm.** Vamos a modificar dicho archivo, por lo tanto modificaremos nuestra copia, la que hemos copiado en **c:\loveasm.**

Ahora nos vamos al W32Dasm, Menu Disassembler --> Open File to Dissasemble y seleccionamos c:\loveasm\regedit.exe



Inmediatamente, el W32Dasm comenzará a desensamblar el fichero. Cuando haya terminado, nos encontraremos ante una pantalla que nos mostrará todas las interioridades del ejecutable. Como veis no sólo hay instrucciones sino también datos, recursos de Windows, referencias a librerías y varias cosas que el programa necesita para funcionar.

Como ya os he dicho antes, el mensaje que nos muestra el regedit cuando se niega a ejecutarse será crucial para crackearlo. El **W32Dasm** nos permite ver las cadenas de un ejecutable o buscar en el listado las referencias a dichas cadenas.



Desensamblado del regedit. Más adelante llegaremos a entenderlo casi todo.

¿Qué es eso de cadenas y referencias a cadenas? Por cadenas me refiero a cadenas de caracteres, es decir, mensajes u otros textos que el programa necesita utilizar mientras se está ejecutando. Y las referencias a cadenas son instrucciones que hacen referencia a esas cadenas.

Pongamos un ejemplo. Supongamos que un programador quiere proteger un programa para que no funcione después de cierto tiempo. Podría escribir algo como esto: (Lo voy a poner en pseudos-código para que todo el mundo lo entienda, sin tener que conocer ningún lenguaje de programación):

```
Si el tiempo ha expirado
Entonces
  Mostrar "Se acabó el asunto, colega"
  Terminar el programa
Sino
  Resto del programa
Fin
```

Aquí aparece una cadena: "Se acabó el asunto, colega". Y una referencia a la cadena en la instrucción "mostrar", porque

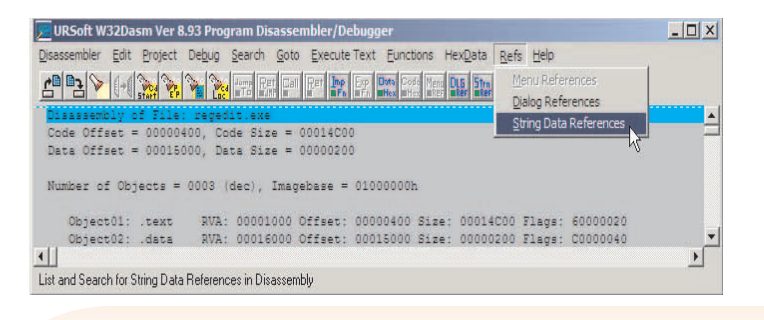
esa instrucción utiliza la cadena para mostrarla en la pantalla. La cadena estará guardada en algún lugar de la memoria (...o del ejecutable) y la instrucción "mostrar" tomará como parámetro la dirección de memoria en la que está la cadena. Ésa es la referencia. De esta manera, si en otro lugar del programa necesita utilizar la misma cadena, no es necesario que la guarde dos veces en la memoria, simplemente volverá a colocar la dirección de memoria en la que está y, de esta manera, habrá otra referencia a la misma cadena.

De modo que, en nuestro caso, tenemos una cadena. Si buscamos las referencias a esa cadena, encontraremos todas las instrucciones que utilizan esa cadena para algo.

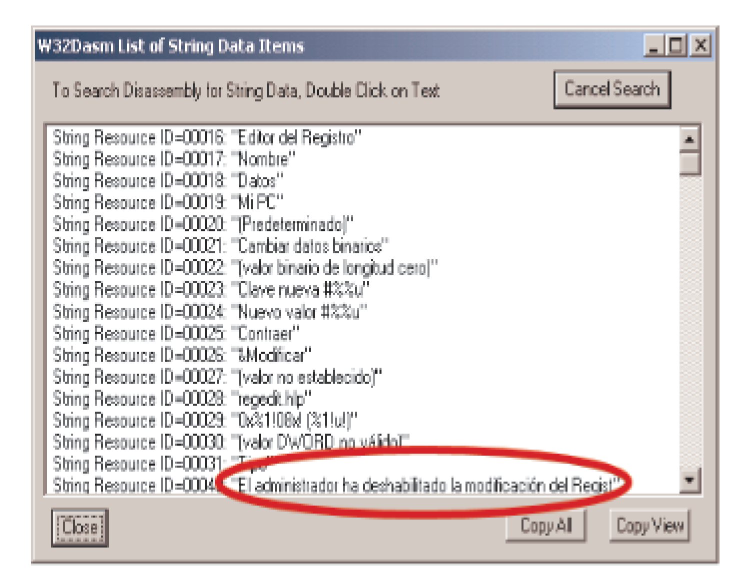
¿Y de qué nos va a servir eso? Si os fijáis en el pseudo-código que he puesto más arriba, veréis que justo encima de la instrucción que utiliza la cadena hay un "si". ¿Qué pasaría si quitásemos ese "si" e hiciésemos que pasara directamente al "resto del programa"? Pues que se acabó la protección.

4. De la teoría a la práctica

Así que vamos a llevar esto a la práctica. En el **W32Dasm** veremos un menú que pone **Refs**. Efectivamente, lo habéis adivinado, ahí es donde están las opciones relacionadas con las referencias. Elegimos Menu Refs --> String Data References

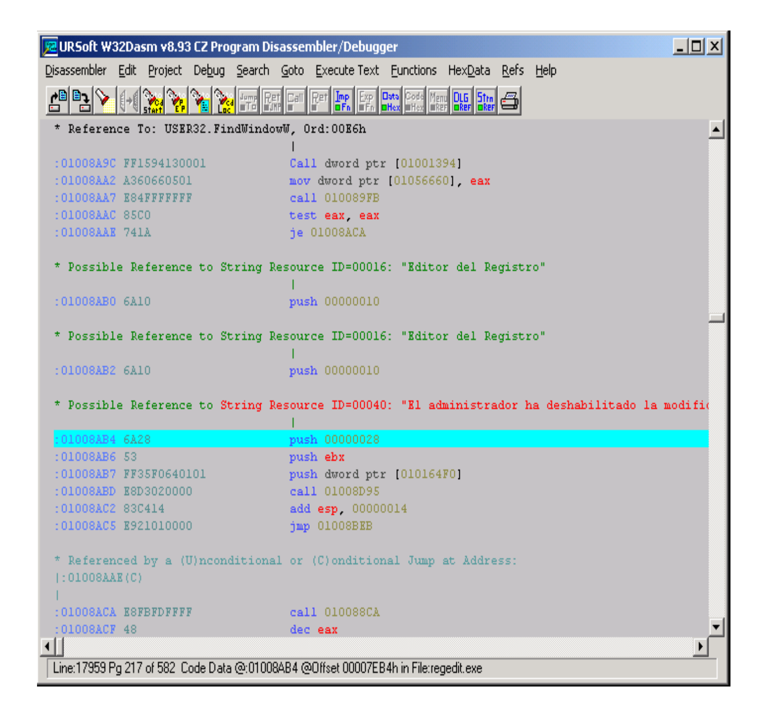


y veremos una ventana que nos muestra una lista de las cadenas que el regedit utiliza internamente. Si nos fijamos un poco, abajo del todo está **la nuestra** (si no la veis, darle un poco para abajo),



Así que hacemos **doble click sobre ella** y aparecemos en una instrucción en medio del listado ensamblador del **regedit**. Si hay varias referencias a la cadena, cada vez que hagamos doble click, apareceremos en la siguiente referencia.

Lo que estáis viendo ahora mismo es un trozo del desensamblado del programa **regedit**. Como veis, está dividido en tres columnas. Cada línea es una instrucción, tanto en lenguaje máquina como en ensamblador. La primera columna es la dirección de memoria en la que está la instrucción, la segunda columna es el código de la instrucción en lenguaje máquina y la tercera columna es la instrucción en ensamblador.
También hay otras líneas con mensajes del W32Dasm que nos dicen cuales son las referencias y qué instrucciones están referenciadas desde otras. Esto lo usan las instrucciones de salto para saltar a otras partes del programa. Vamos a ver lo que significa todo esto.



Si os fijáis un poco más arriba de la referencia, hay una instrucción que empieza con “je”. Esto es un salto. (je = **j**ump if **e**qual, saltar si es igual). Lo que estáis viendo ahora mismo es un esquema similar al pseudo-código que he puesto antes, y tiene la siguiente estructura:

```
comprobar una condición
test eax, eax
je Continuar
mostrar mensaje y salir
continuar:
Resto del programa
```

Las instrucciones clave son **test**, **je** y la **etiqueta**. Lo primero que hay es código para comprobar la protección. Son las instrucciones que hay delante de test. Esas instrucciones dejan un valor en **eax**, que es un registro del procesador, algo así como una variable que contiene un dato.

eax valdrá 1 si la protección esta activada y 0 si no lo está. La instrucción “**test eax, eax**” comprueba si **eax** es igual a cero, y la instrucción **je** hace que, en caso de que **eax** valga cero, el programa siga ejecutándose a partir de la etiqueta continuar. En caso contrario, seguirá por las instrucciones que hay detrás del salto, que muestran el mensaje y terminan.

Así que, lo que tenemos aquí es un esquema igual al del pseudo-código. El “si” es el salto y el parámetro del salto es la dirección de memoria de la instrucción a saltar (como nuestra etiqueta continuar). Podéis ver que esa instrucción está después de la que hace referencia a la cadena. En este caso se trata de un salto condicional (salta SI es igual). Lo que hay que hacer es convertirlo en un salto incondicional, o sea, que siempre salte :)

5. El truco del almendruco

Aquí vamos a meternos un poco con el código máquina, porque lo que vamos a hacer es modificar el código de la instrucción de salto (ese que pone “74 1A”) para convertirla en un salto incondicional.

El W32Dasm no nos permite modificar directamente las instrucciones en ensamblador, lo cual tampoco nos interesa, porque ya os habréis fijado en que no todas las instrucciones tienen la misma longitud. Por ejemplo, la instrucción de salto puñetera ocupa 2 bytes, pero la instrucción **call** que hay un poco más arriba ocupa 5 bytes.

Imaginaros lo que pasaría si nos equivocásemos y cambiásemos una instrucción que ocupa 2 bytes por otra que ocupa más. Sobrescribiríamos las instrucciones que hay a continuación y Windows cantaría su canción favorita: “El programa ha efectuado una operación no permitida”.

Así que tenemos que cambiar el salto condicional de 2 bytes por un salto incondicional que también ocupe 2 bytes.

A parte de saltos condicionales e incondicionales también existen saltos largos y saltos cortos. Se diferencian en lo que ocupan. Aquí están sus códigos:

Saltos cortos:

74 XX	Je XXXXXXXX	Saltar si es igual
75 XX	Jne XXXXXXXX	Saltar si no es igual
EB XX	Jmp XXXXXXXX	Saltar siempre

Saltos largos:

0F 84 XX XX XX XX	Je XXXXXXXX
0F 85 XX XX XX XX	Jne XXXXXXXX
(90) E9 XX XX XX XX	Jmp XXXXXXXX

Por supuesto, las XX del código máquina dependen de las direcciones de los saltos, pero eso no lo vamos a tocar, sólo hay que conocer los números del principio.

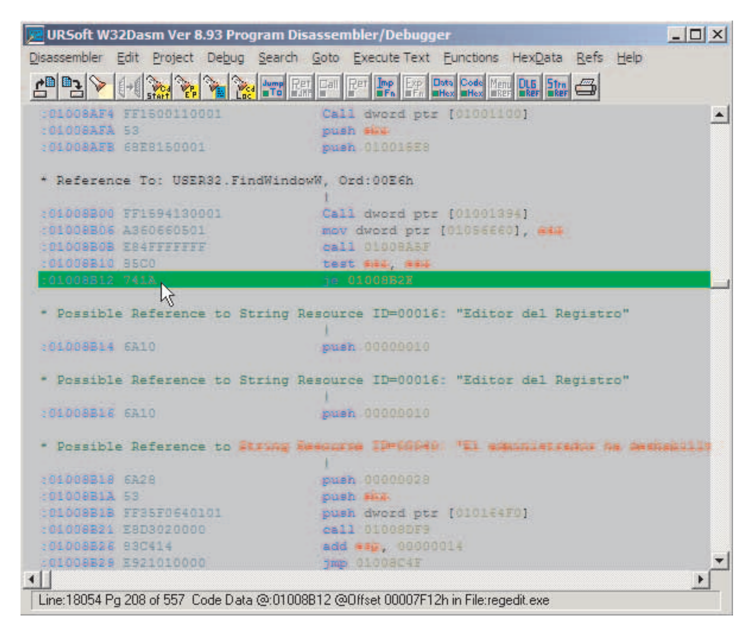
Hay muchos más saltos, pero de momento estos son los únicos que vamos a necesitar. También es interesante conocer una instrucción más. Os la voy a presentar: Se llama **nop**, su código es 90 y sirve para no hacer nada.

A primera vista no parece una instrucción muy útil, pero en realidad es una de las que más se utilizan a la hora de **crackear**. Por ejemplo, si os fijáis en el código largo de la instrucción **jmp** veréis que hay un 90 delante.

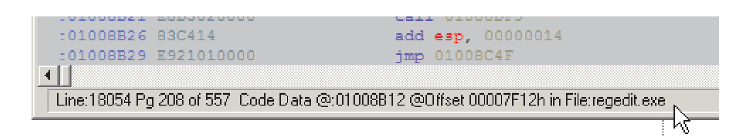
Lo que pasa es que la instrucción **jmp** ocupa un byte menos que los demás saltos largos y para ajustar la longitud nosotros añadiremos una instrucción **nop** delante. Por eso lo he puesto entre paréntesis. Realmente el 90 no forma parte del código de **jmp**.

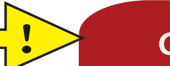
Bueno, pues a estas alturas seguro que ya sabéis lo que hay que hacer. Hay que cambiar el "**74**" por "**EB**".

Primero haced doble click sobre la instrucción de salto para seleccionarla (aparecerá una barra verde sobre ella).



Ahora si os fijáis en la barra de estado del W32Dasm (la de abajo del todo de la ventana) veréis que, entre otras cosas, aparece el texto "**@Offset _00007F12h_ in File: regedit.exe**". **<u>Tomad nota de este número</u>** porque es el offset en el que está el salto dentro del ejecutable (que no es lo mismo que la dirección de memoria).

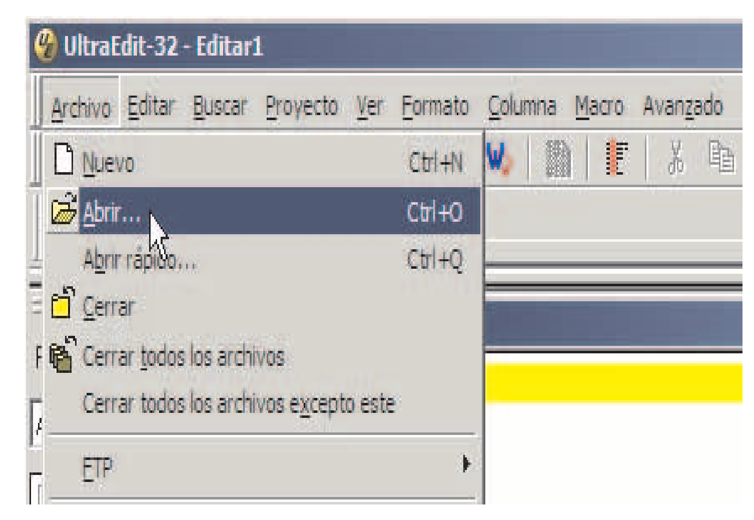




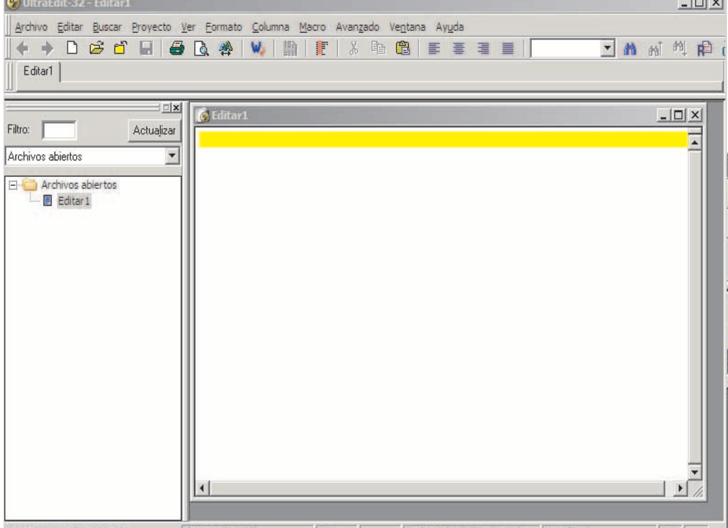
Como ya os dije antes...

Como ya os dije antes, SIEMPRE debéis trabajar sobre una copia de los ficheros originales que vayáis a modificar. Nosotros ya estamos trabajando sobre una copia. **No vamos a modificar el regedit.exe original**. De esta manera, si no tenéis permisos para modificar el regedit.exe podéis hacerlo de todos modos.

Abrid la copia del **regedit** con un editor hexadecimal. ¿Cómo? ¿Nunca has utilizado uno? Bueno, vale… Nosotros vamos a utilizar el **UltraEdit-32** en su versión 10.10c (aunque podrías utilizar cualquier otro).



Lo primero, como siempre, nos vamos a www.hackxcrack.com y descargamos el programa. Lo instalamos y ejecutamos. Os encontrareis frente a la ventana principal del programa :)



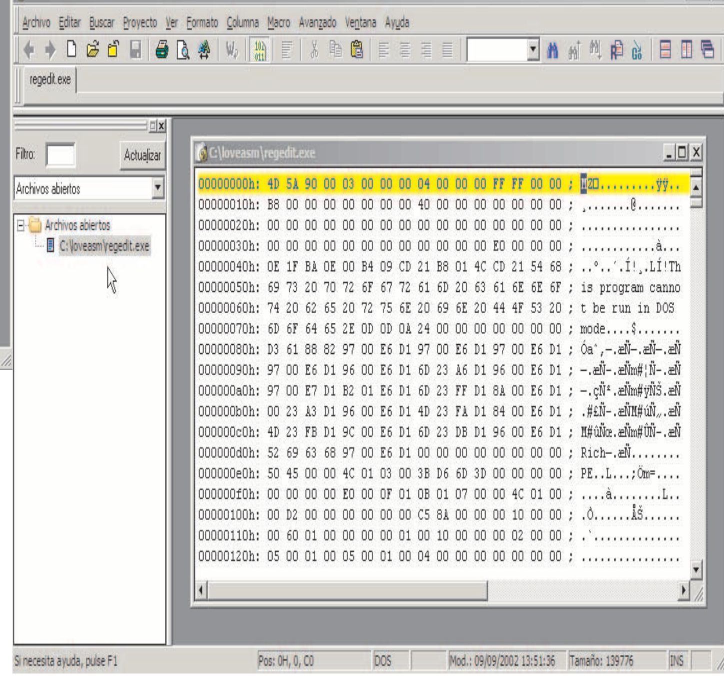
Ahora, desde nuestro nuevo juguete (el UltraEdit-32) hay que ir al offset que decía el W32Dasm (***00007F12h***).

¿Qué como se hace eso?

Bueno, vengaaaa. Ya tenemos el UltraEdit-32 ejecutado y esperándonos, bien. Lo que queremos hacer es EDITAR (modificar) el **regedit**, pues tenemos que decirle al UltraEdit-32 que abra el **regedit**.

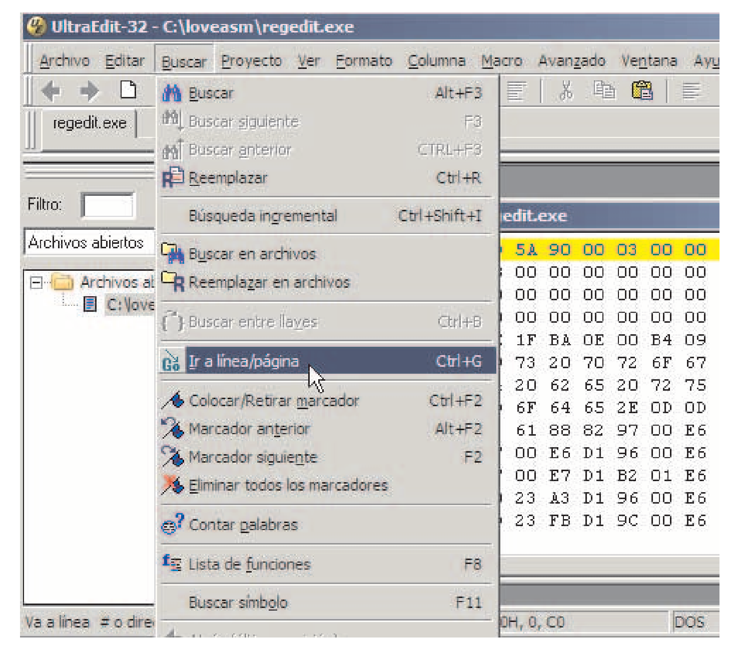
Pues venga, Menu Archivo --> Abrir y le conducimos al **regedit.exe** que tenemos en **C:\lovease**

Como es muy obediente, lo abrirá y tendremos lo siguiente :)



Ahora tenemos el regedit.exe a nuestros pies, podemos editar (modificar) directamente su código. Primero (como ya hemos dicho) tenemos que ir al offset que decía el W32Dasm (00007F12h).

Pues venga, Menu Buscar --> Ir a línea/pagina o la combinación de teclas Ctrl-G

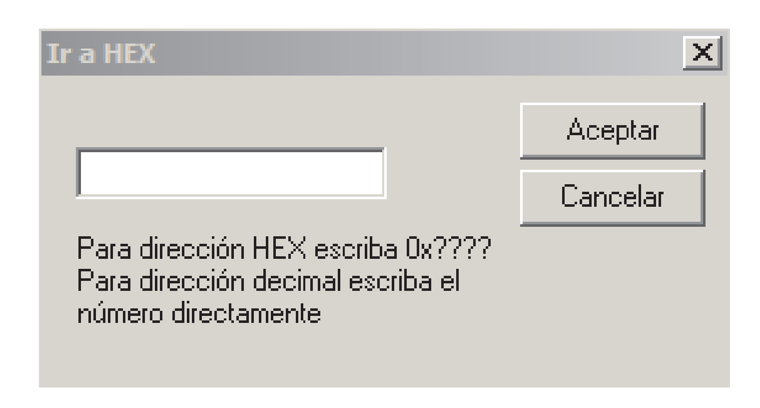




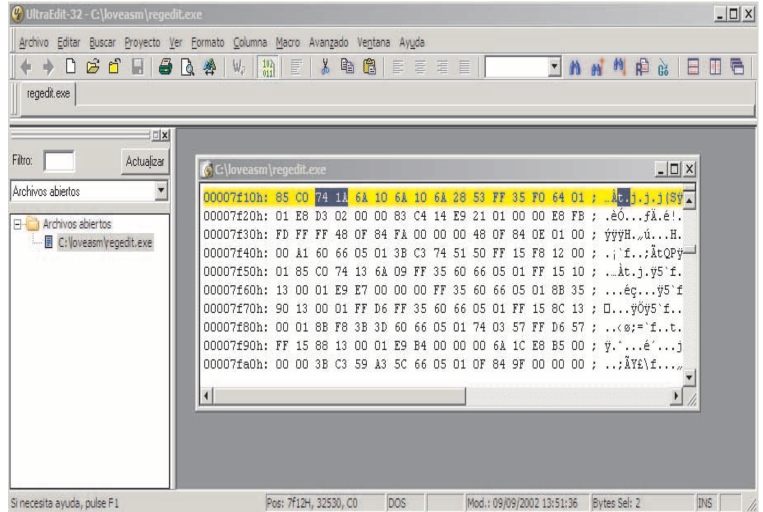
Para los que...

Para los que no utilicen el **UltraEdit-32**:
Como ya hemos dicho, editores hexadecimales hay muchos. Para hacer lo mismo en el **Hex Workshop**, es con **Edit/Goto**... o **Ctrl-G** y debéis seleccionar "**Beginning of file**" y "**Hex**". Usando el **WinHex** es con **Position/Go to Offset**... o **Alt-G**. Aseguráos de que esté seleccionado "**Beginning of file**" y "**Bytes**".

Y nos aparecerá esta escueta ventana:



En el rectángulo blanco hay que escribir el offset **PRECEDIDO** de "**0x**" (el número 0 y la letra x sin las comillas). Es decir, debemos escribir "***0x00007F12h***" (sin las comillas). Si alguien se pregunta el motivo de añadir "**0x**" el mismo recuadro ya lo aclara, estamos introduciendo un dato en HEX (más detalles consultar anteriores número de la revista o consultar en el foro www.hackxcrack.com). Venga, que acabamos, pulsamos el botón aceptar y nos encontraremos lo siguiente:



Lo primero es asegurarnos de que en esa posición está el salto, hay que asegurarse de que estén los códigos "**74 1A**". Fíjate en la imagen, hemos "pintado" de azul los códigos. Si no estuviesen, es que te has equivocado en algo, revisa los pasos a seguir e inténtalo de nuevo.
Una vez hemos encontrado el salto "**74 1A**", deberemos... ALTO!!! LEE LA SIGUIENTE NOTA ANTES DE CONTINUAR!!!



Para los que...

LEGAL o ILEGAL: Ahora te diremos que para seguir el ejercicio "DEBERÍAS" hacer tal y cual cosa. Remarcamos lo de DEBERÍAS, además no te enseñaremos NINGUNA ventanita que indique que nosotros estamos haciendo el ejercicio… ¿por qué?... Pues porque es ILEGAL emplear la ingeniería inversa para MODIFICAR programas de

terceros (en este caso el regedit.exe, un programa de MICRO$OFT).

Por lo tanto, te lo advertimos, si sigues el ejercicio estás cometiendo un delito. Si estuviésemos tu y yo sentados en tu casa te diría que no pasa nada (no creo que en tu casa Bill Gates tenga una cámara espiándote), pero como estás leyendo una publicación legal debemos avisarte y remarcar este tipo de cosas.

Ahora DEBERÍAS cambiar el **74** por "**EB"** (como si estuvieses en Word :). No te enseñamos la pantalla porque es ILEGAL. Y DEBERÍAS guardar el fichero, en el UltraEdit-32, Menu Archivo --> Guardar.

Se acabó!!! Ahora DEBERÍAS ejecutar el regedit.exe que DEBERÍAS haber modificado y que lo tienes en c:\loveasm\. PODRÍAS comprobar que ahora SÍ DEBERÍAS poder acceder al registro de Windows.

6. Ultimando detalles

Si no os funcionase, puede ser porque estáis usando otra versión del **regedit** y la referencia correcta no era esa (el salto correcto no era ese). Ningún problema, borrad la copia del **regedit** y volved a copiar el original en **c:\loveasm**.

Con el **W32Dasm** volved a buscar la referencia. Esta vez, en lugar de una, haced **doble click dos veces** sobre la cadena "**El Administrador ha deshabilitado la modificación del Regist**" en la ventana "**W32Dasm List of String Data Ítems**" (recuerda que llegaste a esta ventana mediante el Menu **Refs** del **W32Dasm)**.

Haciendo esto iremos a la segunda posición y deberemos repetir el resto del proceso. En este caso la referencia del offset será distinta (recuerda apuntarla).

Si con la segunda tampoco funciona seguid probando hasta encontrar la buena. No os preocupéis, no hay muchas. Por ejemplo, en Windows Millenium, un pajarito me dijo que la buena es la segunda ;)

7. Deja el sistema operativo como estaba

Si ejecutáis el regedit.exe original de Windows (situado normalmente en C:\Windows\) veréis que aún sigue mostrando el mensaje **"El Administrador ha deshabilitado la modificación del Registro"**; pero si ejecutases **regedit.exe** que has crackeado (el que tienes en c:\loveasm), funciona sea cual sea el valor que tenga en el registro **DisableRegistryTools**.

Ahora ya podéis borrar **DisableRegistryTools** o poner su valor a 0 (dejar vuestro sistema como estaba).

Con todo esto **DEBERÍAMOS** haber conseguido un **regedit.exe** que PODRÍAMOS llevar a cualquier sitio (por ejemplo un "cyber-cafe") y nos permitirá editar el registro de cualquier PC de la sala aunque el administrador haya deshabilitado esa opción ;)

Ya sabes que el registro de Windows es el "alma" de Windows, si puedes acceder a él y modificarlo puedes hacer "casi cualquier cosa".

Espero que no os haya parecido muy difícil y que lo hayáis entendido bien. Saludos a todos.

CURSO DE SEGURIDAD EN REDES - IDS

- Conoceremos SNORT: Un potente IDS, un potente snifer, un potente "caza-intrusos"
- Sistemas IDS: HIDS, NIDS y DIDS
- Vamos a acercarnos a LA VERDADERA SEGURIDAD!!!

Hola **amig@s**, otra vez por aquí ;)

En esta ocasión no vamos a probar vulnerabilidades, todo lo contrario, vamos ha intentar taparlas, protegernos y asegurar nuestros PC's. Y cuando digo nuestros PC's digo TODOS, no sólo uno de ellos.

Este es el primero de una colección de artículos que tratan de **SEGURIDAD**, lo pongo en mayúsculas porque eso es lo que vamos a implementar, ***seguridad, seguridad y seguridad***.

La idea esencial es aprender dos cosas básicas:

- Un Sistema de Detección de Intrusos (**IDS - Intrusion Detection System**)
- Un Cortafuegos (**Firewall**) y/o **Proxy-Firewall**

Empezaremos por los IDS. Cuando me decidí a escribir sobre estos temas me surgió una gran duda entre dos posibilidades: *¿Qué haces, Vic_Thor? ¿Pones cuatro links, das tres ejemplos y te ventilas el IDS en dos páginas? O.... ¿Explicas el cómo, por qué, dónde y de qué manera se implementa un IDS?*

Opté por la última, lo que quiere decir que son *"malas y buenas"* noticias, *"malas"* porque me tendréis que aguantar más de un mes ya que me decidí a explicar pormenorizadamente la arquitectura de ***snort***, que será el ***IDS*** a implementar, y serán *"buenas"* porque tras esta serie de artículos seréis capaces de manejar sin problemas, sin miedos y con total seguridad este tipo de aplicaciones.

Algunos estaréis pensando... *"Joer, qué misterio tendrá... si total se instala, se aplican las reglas, se ejecuta y a mirar..."* bueno, pues si queremos entender... primero hay que aprender... además ya conocéis la filosofía de esta publicación: *"... no sólo nos gusta que funcione, nos gusta saber cómo funciona..."*

Si tienes un conocimiento medio de ***snort***, este artículo te sabrá a poco, pero si no sabes lo que son cosas como el *pre-procesador, Frag2, Stream4 ó Stream5, el sistema de detección, plug-in para salida, etc...* este artículo te abrirá los ojos y te ayudará a entender *"el alma"* de los **IDS** y sobre todo ASEGURAR la red mediante un sofisticado mecanismo de alertas y detección de ataques comunes, exploración de puertos, exploits, desbordamientos de memoria, etc.

Terminaremos creando un **Sistema Central IDS**, con soporte de Bases de datos, servicios Web para la generación de informes, alertas por mensajería, reglas personalizadas y más....

Las aplicaciones escogidas se ejecutan sin problemas en plataformas *LiNUX y Windows*, quizás en lo que más difieren unas versiones de otras es en el momento de la instalación, de los *"añadidos"* y de alguna que otra cuestión menor o mayor dependiendo de cómo lo veamos.... con gran esfuerzo y cariño hacia *vosotr@s* estos artículos están pensados en ambas plataformas, lo iréis descubriendo poco a poco.

En artículos posteriores abordaremos ***firewalls y proxys*** no... *no se trata de ZoneAlarm... que va...* es algo más profundo y *"serio"*,

flexible y configurable, que analice no sólo puertos e Ip's, también contenidos y, si nos animamos, hasta nos terminamos montando un *Honeypot* para despistar a nuestros *"atacantes"*, me estoy refiriendo a ***Firewalls del tipo Next Generation o Firewall 1 de Checkpoint, a squid o ISA Server...***

No me gustan las dedicatorias, pero en este caso se merecen... este artículo se lo dedico especialmente a mi hijo ***Oscar***, que con sus siete años *me ha tenido que aguantar demasiados fines de semana "enfrascado"* en el ordenador mientras él jugaba o veía sus *"pelis"* o me "daba *la lata"* con eso de.... *"Papa y ahora qué hacemos.... deja ya de trabajar...."*

Mil besos, hijo por tu paciencia y tu cariño.

Alarma, Alarma... Intrusos !!!!

Parece obvio y casi todos podríamos dar varias definiciones de lo que es una ***intrusión***, yo me voy a quedar con dos:

Intrusión es el acto de entrar en un lugar sin invitación o sin ser bienvenido, *es como colarse en una fiesta privada a la que no fuimos invitados.*

En el mundo de las redes, una ***intrusión*** es simplemente el intento de comprometer cualquiera de nuestros dispositivos de red, ya sean *switches, routers*, estaciones de trabajo, servidores o cualquier otro medio que brinda conectividad o se conecta a la red.

Visto de ese modo, un ***IDS*** podría compararse con una alarma antirrobo, detecta las intrusiones o accesos no permitidos y hace *"sonar la sirena"*.

Lo cierto es que un ***IDS*** hace algo mas... reconoce a los intrusos, los identifica e incluso puede parar el ataque, de este modo, un ***IDS*** es como una alarma antirrobo con cámaras de seguridad, junto con un banco de datos que reconoce al intruso y con un sistema de vigilancia que registra los accesos y envía "la *policía"* a detener al ladrón.

Una de las mejores analogías para describir un ***IDS*** es la de un antivirus... estos examinan los archivos de tu disco duro en busca de códigos dañinos, establecen un sistema de protección de archivos, memoria y arranque, y protege a los archivos en tiempo de ejecución.

Esto mismo es lo que hace un ***IDS***, pero en lugar de inspeccionar archivos, particiones y bloques de memoria, analizan paquetes de datos, analizan el tráfico de la red en busca de patrones predefinidos que hagan *"sospechoso"* un paquete de datos a nivel IP o MAC.

Al igual que sus *"primos"* los *firewalls*, pueden ser Software, Hardware o una combinación de ambos, y pueden detectar accesos no autorizados desde dentro o fuera de la red.

Aunque un ***IDS*** puede configurarse en el mismo sistema donde corre un *firewall*, un servidor, etc.. no es muy recomendable, lo más habitual es que el ***IDS*** sea un equipo independiente puesto que la carga de procesador puede ser muy alta si la red es muy *"parlanchina"*.

Un ***IDS*** puedes imaginarlo como un análisis de sangre, por el simple hecho de hacerte un chequeo no curarás una enfermedad, un ***IDS*** hará saltar las alertas, *avisarte de que estas bajo de hierro, que tienes un exceso de colesterol o que los trasaminasas están por las nubes*, *"curar"* la enfermedad pueden escapársele al ***IDS***, *puede parar la ingesta de alimentos que aumenten el colesterol o que dañen el organismo pero realmente no cura la enfermedad...*

Para lograr estas proezas, utilizan diversas técnicas, desde la *"detección de firmas"* como lo hacen los antivirus, pasando por las huellas de ataques conocidos hasta las detecciones *"anómalas"* de tráfico inusual, actividad *"normal"* o *"anormal"* muy parecido a como lo hacen los antivirus con el llamado análisis heurístico.

Algunos ***IDS*** pueden hasta llegar a desconectar el *host* atacado, incluso aunque fuese un *router* o la puerta de enlace, para mantener *"el orden"* y la seguridad en la red.

Los ***IDS*** funcionan como el análisis forense de la escena de un crimen, obtienen las huellas dactilares, análisis de ADN, pautas de comportamiento, muestras de cabello, sangre, etc... de ese modo recomponen *"el ataque"* lo identifican, lo detienen o al menos lanzan las alertas.

IDS hay más de uno...

Aunque es habitual hacer referencia al ***IDS*** como el ***Sistema Detector de Intrusos***, ciertamente puede ser más sofisticado que una máquina perdida por nuestra red.. disponemos de tres tipos de ***IDS***, los ***HIDS, NIDS y DIDS***, los vemos a continuación:

- ***HIDS (HostIDS)*** esto es**, un IDS vigilando un único *host*** que observa únicamente a ese *host*, en este caso, **la tarjeta de red *corre* en modo no promiscuo**, es decir, sólo recogerá, analizará y detectará paquetes de datos que vayan dirigidos a ese *host* en concreto y los paquetes que salgan de ese *host*... bueno algunos otros también...

  Su mayor ventaja reside en un uso menor de la *CPU* y en la existencia de tarjetas de red que por hardware no es posible ponerlas en el modo *promiscuo*.

  Otra ventaja es la de *"acotar"* las reglas de procesamiento de los paquetes a analizar, por ejemplo, si instalamos un ***HIDS*** en un servidor web, podemos eliminar del conjunto de reglas ataques a otro tipo de servicios y servidores como DNS, FTP, etc.



Modo Promiscuo...

Modo Promiscuo / Modo No Promiscuo: Si te estás preguntando cómo poner una Tarjeta de Red en modo Promiscuo, tranquilo, cuando lleguemos a la parte practica del artículo lo podrás hacer tu mismo :)

En la ***figura 1*** observamos una red con 4IDS basados en *host*, uno para el Servidor Web corporativo y otros tres para equipos individuales de la LAN.

Figura 1. IDS basado en Host. HIDS

- ***NIDS (NetworkIDS)*** o ***IDS basados en red***. Este tipo de sistemas detectarán ataques a TODO el segmento de la red en el que corre el *IDS*, **la tarjeta de red del *NIDS* debe corren en modo promiscuo**, es decir, capturará TODO el tráfico que vaya destinado a cualquier equipo del segmento de red y no sólo al equipo donde se ejecute el ***NIDS***

La utilización de este sistema debe ser autorizado por el administrador y monitorizado, puesto que teóricamente puede resultar comprometida la red completa. Incluso pueden utilizarse más de un ***NIDS*** para asegurar con mayor efectividad todos los *host*.

En la *figura 2*, vemos un sistema *IDS* basado en dos ***NIDS*** el primero protegería el segmento compuesto por el Servidor Web y el Servidor Mail, mientras que el segundo protegería a

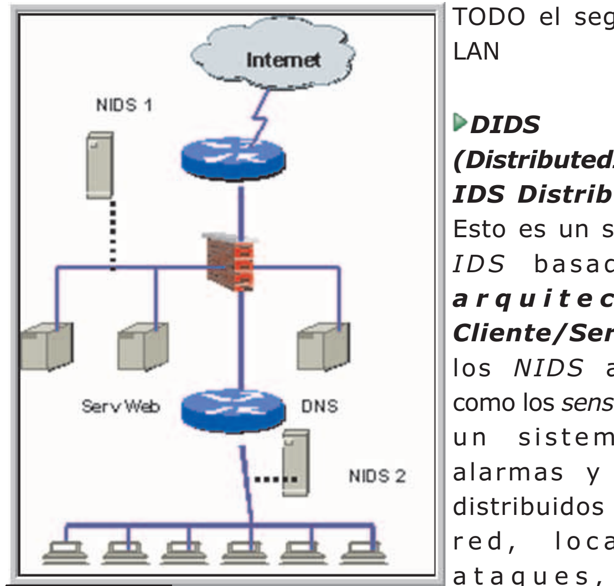


Figura 2. IDS basado en red. NIDS

TODO el segmento LAN

DIDS (DistributedIDS) o IDS Distribuidos. Esto es un sistema *IDS* basado en ***arquitectura Cliente/Servidor,*** los *NIDS* actúan como los *sensores* de un sistema de alarmas y están distribuidos por la red, localizan ataques, los centralizan y pueden almacenar o recuperar los datos de una Base de Datos centralizada.

En este modo, ***las tarjetas pueden estar en modo promiscuo o no promiscuo***, las reglas de cada una se hacen *"a medida"* según las necesidades de cada segmento, *host* o red completa, las alertas, las huellas, las firmas de ataques, los registros, etc., se extraen o vuelcan en un sistema central, este tipo de soluciones son habituales en arquitecturas del tipo *VPN* (*Virtual Private Network, Redes Privadas Virtuales*)

Un ***DIDS*** puede ser un sistema de *NIDS*, *HIDS* o una combinación de ambos, son soluciones complejas con alto grado de seguridad.

En la *figura 3*, tenemos un ***Sistema Distribuido IDS***, en la que los 3 *NIDS* de red se comunican con el *NIDS* central dónde se almacenan las incidencias, *logs*, etc...

El uso de un *IDS* con *firewalls y/o routers* con listas de acceso (*ACL's*) implementadas que permiten o deniegan el tráfico eleva al máximo la seguridad en la red, en estos casos un *IDS* se convierte en un analizador de los cambios en el comportamiento del *firewall* o de las *ACL* de y proveen a los mismos de un mecanismo de auditoria y control.

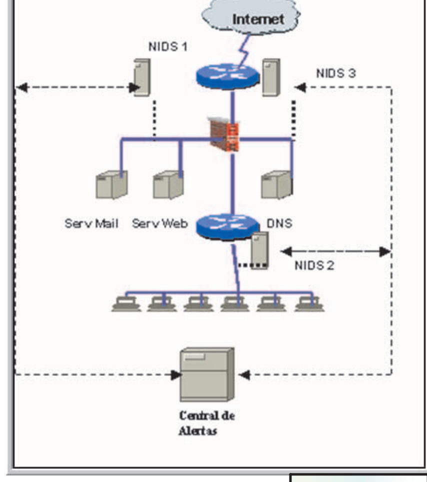


Figura 3 IDS Distribuido

Te estarás preguntando ... *¿Entonces si uso un IDS puedo pasar del firewall?* **R:NO**, el *firewall* bloquea los accesos a puertos no permitidos y generalmente cumple bien sus cometidos, pero en muchos casos no están diseñados para analizar el contenido de los paquetes que circulan por los puertos autorizados.

El motivo por el que los *firewalls* (y sobre todo los personales, tipo Zone Alarm, Kerio, etc..) no analizan los paquetes es por el alto consumo de recursos. El uso de ciclos de CPU es extremadamente intenso y el rendimiento del equipo sería muy bajo, no digamos nada si se trata de un *proxy-firewall*, deberíamos disponer de un *"maquinón"* para soportar en la misma máquina un *proxy*, un *firewall* y un *IDS* corriendo a la vez.

Además, por los puertos permitidos puede correr tráfico *"enmascarado"*, ya sabes, lo de *tunelizar* las conexiones y hacer pasar algo por lo que no es. Imagina un servidor de *IRC*, Web, *KaZaA* o un servidor de mensajería instantánea, es posible configurar un *IDS* para detectar e informar de tráfico no "habitual" por esos protocolos y puertos.

Un *IDS* puede ser configurado para monitorizar una red, pasiva o activamente. En el modo pasivo se limitará a registrar los accesos y a guardar *logs* de los mismos, en el modo activo puede reaccionar contra la intrusión.

Se pueden monitorizar Servidores de Bases de datos *(SQL, MySQL, ORACLE...)* Servidores *DNS*, Servidores Mail, Servidores de administración remoto (*VNC, TS, Citrix, PcAnywhere...*)

Snort, El espía que me amó....

El creador de ***Snort*** fue ***Marty Roesch*** y en un principio lo llamó *"Lightweight Intrusión Detection System"*, (Sistema ligero de detección de intrusos), sin embargo, ***snort*** es todo menos *"ligero".*

Snort es capaz de analizar en tiempo real el tráfico de paquetes en un equipo o en una red corporativa y además mantener un registro de *logs* e identificar ataques concretos en función de su *"huella dactilar"*.

Snort puede ejecutarse en multitud de Sistemas Operativos, es una distribución *"Open Source"* y bastante común entre los administradores de sistemas.

Si lo deseamos ***snort*** puede ser un *"simple" esnifer*, es decir, podemos configurar ***snort*** sin reglas de procesamiento y análisis de tráfico. Aunque lo normal es que no sea así, una de las más poderosas características se ***snort*** es la de utilizar registro de sucesos, y métodos de alertas volcando sus resultados en servidores *Syslog* ya sea en texto plano, archivos XML o incluso enviar mensajes del tipo *WinPopup* a clientes *Windows*.

Según las definiciones que hemos ido dando anteriormente, ***snort*** es una vacuna para los paquetes de datos que circulan por la red, es un sistema de Rayos X que inspecciona estos paquetes, los captura (***sniffer***) los coloca en un contenedor o registro (***packet logger***) y/o los examina en busca de patrones, ataques y análisis forense (***NIDS***)

Snort* es un sistema *NIDS, puede analizar tráfico IP en busca de *"huellas"* comunes de un ataque, analiza los protocolos individualmente, como ejemplo puede detectar ataques del tipo *Buffer overflow*, ataques *CGI, SMB, RPC* o un escaneo de puertos.

Podemos configurar ***snort*** en tres modos básicos:

- ***Sniffer***, como un ***"simple" analizador de protocolos***
- ***Packet logger***, como un ***capturador y "grabador"*** de paquetes
- ***NIDS***, como un **sistema de detección de intrusos en red**

Lógicamente, configurar ***snort*** como un *NIDS* es la forma más compleja y flexible de usarlo.

Snort se basa en firmas *IDS* y usa reglas para comprobar los paquetes que viajan por la red y genera alertas enviándolas a una Base de datos, un archivo de sucesos o mensajes *SNMP*.

Originalmente fue concebido como un esnifer, ***Martín Roesch*** lo escribió para *LiNUX* y para su propio uso, apenas si contaba con 1.600 líneas de código distribuidas en dos ficheros., tras el éxito obtenido y las sucesivas revisiones del producto, hoy en día se puede implementar en varios sistemas operativos y su código asciende a más de 75.000 líneas., hasta dispone de *plug-ins* para bases de datos como MySQL o Postgres, para RPC, etc.. hasta *plug-ins* gráficos para una administración *"mas sencilla"* . Igualmente existen *plug-ins* para protocolos específicos, como HTTP, 802.11 (wireless) ARP, etc...

Las reglas de ***snort*** incluyen diversos tipos de servicios, *P2P*, *Backdoors, troyanos y/o puertas traseras, Denegación de Servicios DoS y DDoS, ataques Web*, e incluso *algunos virus* conocidos.



Si eres un nuevo lector...

Si eres un nuevo lector de PC PASO A PASO, seguro que te desespera leer cosas como MySQL, Postgres, RPC, HTTP, 802.11, ARP, P2P, Backdoors, DDoS, DoS. No dejes que te atemoricen!!! La mayoría han sido explicados ya en anteriores números, de todas maneras ya verás que podrás seguir este artículo sin necesidad de ellos :)

Algunos *links* interesantes para empezar con *snort:*

http://sourceforge.net/mailarchive/forum.php?forum=snort-users ---> Foros de uso de la aplicación.

http://sourceforge.net/mailarchive/forum.php?forum=snort-sigs ---> Foros dedicados exclusivamente al uso de las reglas.

Y cómo no... http://www.snort.org/ ---> Para encontrar documentación, descargas, plug-ins, etc...

http://www.snort.org/cgi-bin/done.cgi y http://www.snort.org/cgi-bin/needed.cgi --> Índice de reglas.

http://www.snort.org/docs/ ---> Documentación.

http://www.snort.org/dl/ ---> Descargas.

Las reglas se organizan por números (los llamados SID o SIGS), por ejemplo el Sids para los ***ataques a IIS del bug de unicode y otros son 970, 974, 981, 982, 983***, *recuerdas esto de algo, ¿no? (números dos y tres de la revista).*

La última versión de ***snort*** disponible es la 2.1.1 aunque algunas herramientas y *plug-ins* están orientadas a la versión 2.0.x con pocos cambios y revisar el *changelog* de la página oficial para conocer las mejoras implementadas por la versión 2.1.1 podremos usarlos.

Requisitos Hardware y Software para Snort

Hardware

Si configuramos ***Snort*** como *NIDS*, de todos los componentes *Hard* de una máquina, debemos pensar en disponer de un **buen espacio en Disco** para almacenar los *logs*,.

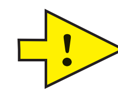
En el ***Disco duro*** se almacenarán los archivos y registros de sucesos, de momento vamos a ir pensando en reservar unos ***10 GigaBytes*** por cada *"sensor"* que instalemos en la red.

El otro componente crítico es ***la tarjeta de red***, o mejor dicho... *"las tarjetas de red...."* **aconsejable dos** una para la escucha ***"pasiva"*** y otra para la conectividad básica del segmento de red.... aunque se puede implementar en una sola tarjeta de red, no es lo ideal.

Por supuesto **la velocidad de la tarjeta de red deberá ser acorde con la velocidad de la LAN**, si instalamos un equipo ***snort*** con una tarjeta de red de 10 Mbs en una LAN con PC's equipados con tarjetas de red a 100 Mbs, perderemos paquetes y tendremos resultados erróneos.

No olvides que ***snort*** es un programa de escucha ***PASIVO***, sólo escucha y analiza... una táctica para *"ocultar"* ***snort*** a ojos y miradas indiscretas es la de utilizar cableado en el que los hilos de transmisión no están conectados (***snort es pasivo,*** sólo escucha) de ese modo si alguien en la red o fuera de ella *"busca"* el *IDS* no lo encontrará ya que esa máquina no responderá al tener *"capada"* su posibilidad de transmitir.

Por lo demás, snort no es muy quisquilloso, aunque como en todas las aplicaciones de alto rendimiento, mejor cuanto más veloz sea el microprocesador, mejor 256 MB de RAM que 128Mb... también dependerá de la plataforma elegida... a *Windows* le gusta mucho la RAM...



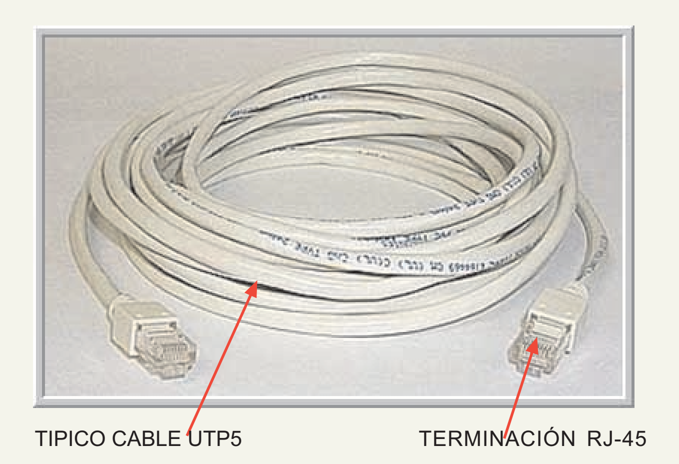
Para los curiosos

Para los curiosos: ¿Qué es eso de utilizar cableado en que los hilos de transmisión no están conectados? ¿eso existe? :)

No es necesario disponer de un cable "capado para transmitir" para seguir este artículo, pero como sabemos que esta revista la compran "mentes MUY curiosas", te invitamos a investigar el tema.

Actualmente, para conectar los distintos elementos de red se utilizan los típicos cables UTP5 y en sus extremos clavijas (enchufes) tipo RJ-45 (parecidos

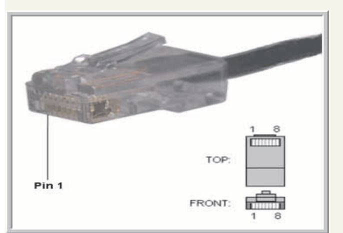
a los del teléfono pero más gruesos). Dentro del cable UTP5 hay 8 hilos y cada uno tiene su misión, unos hilos transmiten, otros reciben, etc.



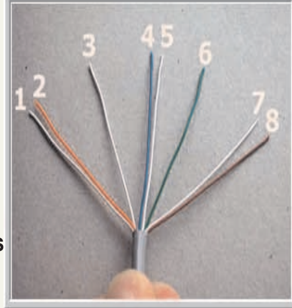
TIPICO CABLE UTP5 TERMINACIÓN RJ-45

Si eres de los que le encanta el PC-Bricolaje, tu mismo puedes hacerte estos cables, consulta en el buscador ***www.google.com*** por "construir cable ethernet" y encontrarás páginas como http://www.coloredhome.com/cable_cruzado/cable_cruzado.htm, donde te explica paso a paso el tema. A medida que investigues, verás que puedes construir cables ethernet "normales" (para conectar los PC al Hubs y Switch) y cables ethernet "cruzados" (para conectar dos PCs sin necesidad de Hub o Switch, directamente de -tarjeta de red- a -tarjeta de red-).

Después del "paseo" por google, estarás en condiciones de crear tu propio cable y crearlo "como tu quieras". En nuestro caso concreto, buscamos construir un cable ethernet "normal" pero sin conectar los hilos 1 y 2 (que son los que transmiten). De esta forma tendrás un cable "normal" pero "capado para la transmisión" (solo podrá recibir datos).



DETALLE DE UN CONECTOR RJ-45
Y LA NUMERACIÓN DE LOS HILOS



Si no tienes ganas de hacerlo tu mismo, siempre puedes ir a una tienda de venta de componentes para PC y pedir que te hagan un "Cable Ethernet UTP5 RJ45 no cruzado" PERO remarcándoles que NO CONECTEN a la clavija RJ45 los hilos 1 y 2. Te dirán que entonces el cable no funcionará correctamente, diles que ya lo sabes, que lo quieres así para que la tarjeta conectada sea incapaz de transmitir PERO si de recibir :)

Ya está, no te damos más la lata.

Software y Sistema Operativo

Snort **corre en la práctica totalidad de Sistemas Operativos modernos**, desde los basados en x86 como ***LiNUX***, **FreeBSD**, ***Windows***, etc.. hasta otros Sistemas basados en **Sparc Solaris**, **PowerPC MacOS**, **RISC HP-UX**...

Las **Librerías libcap** o **WinPcap**

Opcionalmente y dependiendo de la configuración de snort puede ser necesario disponer de:

- MySQL, Postgres, ORACLE o bases de datos SQL
- Clientes SMB si usamos mensajería WinPopup
- Servidor Web Apache u otros como IIS
- PHP, Perl para determinados plug-ins y add-ons
- SSH o Terminal Services para acceso remoto
- Módulos SSL para Apache o IIS

Medios de red. Switches y Hubs

La principal diferencia entre un hub y un switch estriba en cómo se reparten el cable los equipos conectados a los mismos.

Frecuentemente se dice que las redes conectadas con un **HUB son redes compartidas** (Todos los PC's comparten el Ancho de banda disponible) mientras que las

redes conectadas mediante un **switch son redes conmutadas** (Cada Pc dispone de su propio ancho de banda para el sólito).

En redes compartidas TODOS los equipos pueden escuchar el tráfico generado en ese segmento, por tanto un equipo con una tarjeta en modo promiscuo será capaz de escuchar, capturar y analizar "las conversaciones" de cualquier otro.

En redes conmutadas sólo podremos escuchar nuestras propias conversaciones y aquellas que vallan dirigidas a todos los host (el llamado **broadcast**).



TIPICO SWITCH DE 24 PUERTOS (24 "ENCHUFES" RJ45)

Cuando entres...

Cuando entres en una oficina, en un centro oficial del estado o en un hospital, a partir de ahora fíjate en los ordenadores, en los cables y seguro que acabarás encontrando (normalmente en el techo o en una pared) un "Rack de Switches". Estos RACKS conectan TODOS los ordenadores del centro y el aspecto es de unos cuantos switches apilados y un montón de cables ethernet conectados.

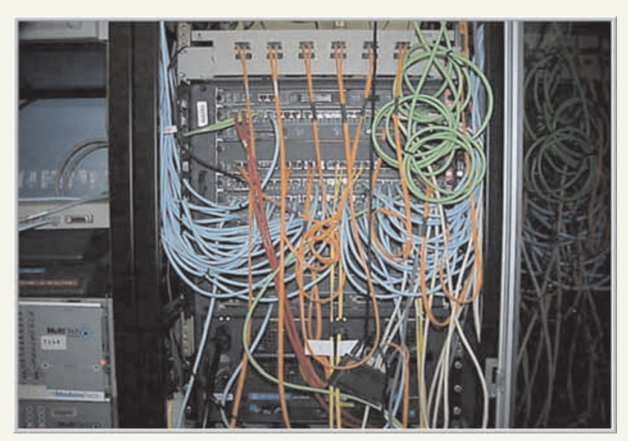


Imagen "bastante extrema" ;) de lo que puedes llegar a ver en algunas oficinas.

IDS y HUBS

Los ***HUB*** repiten la señal por cada puerto excepto por el que ha llegado, es decir, un ***hub*** propaga TODO por TODAS las bocas o puertos disponibles.

Un HUB trabaja en la capa 1 del modelo OSI, trabaja a nivel físico puesto que **su único cometido es amplificar, regenerar y repetir la señal.**

Si conectamos nuestro *IDS* a cualquier puerto de un ***HUB*** y ponemos la tarjeta en modo promiscuo seremos capaces de analizar TODO el tráfico de la red sin problemas.

La desventaja de usar ***HUB*** está en que la velocidad de transmisión entre dos puntos de la red es inferior a si usamos un *switch*.

IDS y SWITCHES

Los ***switches*** se dice que **dedican el ancho de banda** puesto que **la comunicación entre dos host cualquiera de la red no la propaga por otros puertos** que no sean aquellos en el que están conectados ambos.

Un switch trabaja en capa 2 del modelo OSI, a nivel de enlace de datos, **toma decisiones en función de la dirección MAC** de las tarjetas de red y **relaciona esas MAC's con los puertos físicos a los que están conectados en el switch.**

*Por ejemplo, cuando un host conectado al puerto 1 se comunica con un host conectado al puerto 7, el **switch** conmuta en su interior el tráfico y lo hace salir o entrar únicamente por los puertos 1 y 7, el resto de ellos ni se enteran...*

Cuando un *host* quiere comunicarse con TODOS en un ***switch***, propaga una señal *broadcast* dirigida a todos los *host*, en esos casos el ***switch*** se comporta como un *hub*, pero sólo cuando el tráfico de red es *broadcast*.

Si conectamos nuestro *IDS* a cualquier puerto de un ***switch*** y ponemos la tarjeta en modo promiscuo sólo seremos capaces de analizar el tráfico de la red que vaya dirigido a nuestro equipo y/o el tráfico *broadcast* del segmento de red.

Es decir, si no se pone remedio a este asunto *snort* sólo podría ejecutarse como un *HIDS, IDS basado en Host*, independientemente de que la tarjeta esté en modo promiscuo o no.

Para estos casos y algunos otros también, los ***Switchs*** disponen de los llamados puertos *SPAN (Switch Port Analiser, Puerto analizador de switch)*

Un puerto *SPAN* es capaz de escuchar todo el tráfico de los otros puertos, como un *hub*, de forma que un *IDS* conectado al puerto *SPAN* del ***switch*** nos permitirá utilizar *snort* como un *NIDS*.

Todavía se puede complicar más... *VLAN's* , ***switches*** en stack, en cascada, redundantes...bueno, pensemos en un ***switch*** con puertos *SPAN*, aunque lamentablemente no todos los ***switches*** tienen esa capacidad. Si ese es tu caso todavía quedan opciones, pero degradará enormemente la eficiencia de la red, mejor conformarte con instalar un ***snort*** en modo *HIDS* o prepara el bolsillo y cambia de ***switch*** (te saldrá muy caro).

Arquitectura de Snort

Antes de empezar con ***snort*** es útil conocer sus componentes básicos, ya hemos dicho que puede ser un *esnifer*, un generador de *logs* o un *NIDS*, además le podemos añadir un motón de *plug-ins* para personalizarlo , ***snort*** se compone de cuatro componentes básicos:

1.- El esnifer
2.- El preprocesador
3.- El sistema de detección
4.- El sistema de salida (Alertas y logs)

Realmente ***el preprocesador***, ***el sistema detector y la salida son plug-ins*** que se han añadido como parte del código de ***snort***.

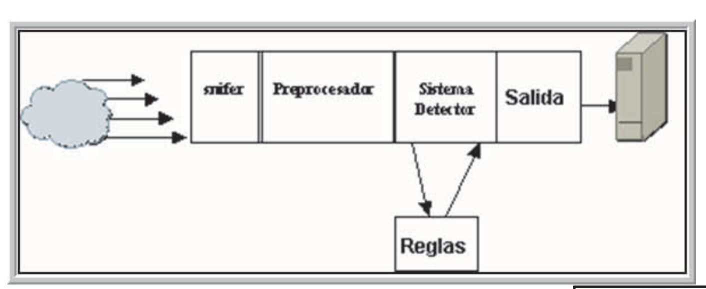


Figura 4. Arquitectura y componentes de snort

El módulo Esnifer

El ***esnifer*** captura los paquetes de datos que circulan por la red, el ***prepocesador*** determina si esos paquetes deben ser analizados y los pasa al ***sistema de detección*** que se encargará de comprobar si existen reglas específicas para ese tipo de paquetes y en su caso las aplica.. Dependiendo de si se cumplen o no dichas reglas se generan alertas, mensajes, logs, etc.. ese trabajo queda destinado para el ***sistema de salida***.

Si hacemos otra analogía.... *pensemos en un cajero automático cuando nos disponemos a pagar el importe del parking...* por la ranura introducimos las monedas o billetes (el ***esnifer***), el ***preprocesador*** dirige esas monedas o billetes por los conductos del cajero hacia el ***sistema de detección*** que analiza cada moneda, billete, etc verificando que son auténticos, que no hemos *"metido otras cosas"* y aplica las reglas de validación. Cuando termina de verificar el dinero emite una salida (***el sistema de salida***) que puede ser el cambio, una sirena de aviso, un mensaje de que faltan x euros o quedarse con el ticket del parking porque le estamos intentando engañar :P

El ***esnifer*** puede analizar tráfico *IP, TCP, UDP, ICMP*, *protocolos de enrutamiento como RIP, OSPF, etc..*

Los ***esnifers*** se pueden usar *"para lo bueno y para lo malo"*, depende de quien lo use. Podemos,. desde detectar problemas de

comunicación (cuellos de botella, etc) hasta capturar contraseñas y datos sensibles...

Otra forma de usar las capacidades de ***sniffing*** de *snort* es volcar o guardar el tráfico capturado en tráfico *"logueado"* para analizarlo más tarde.

El preprocesador

El ***preprocesador*** recoge los paquetes que le pasa el *esnifer* y comprueba si dispone de *plug-ins* para analizar el tipo de paquete, por ejemplo *HTTP, RPC, SMB, Escaneo de puertos, etc.*.

Existen muchos plug-ins para el ***preprocesador*** de ***snort***, sería imposible relatar todos y dar una explicación pormenorizada de cada uno de ellos, más adelante veremos como usarlos.

Figura 5. Funcionamiento del Preprocesador en un ejemplo para RPC

Aunque en los próximos artículos hablaremos más a fondo del preprocesador y el sistema de detección, es importante ahora que conozcas algunos términos del mismo.

Las directivas o reglas del preprocesador se nombran de la forma: ***protocolo_decode*** donde protocolo puede ser un protocolo, servicio o directiva a aplicar.

Hay varios, pero los más interesantes son:

- ***http_decode*** para examinar el flujo de datos en servidores web y/o protocolo http.
- ***ftp_decode,*** para servidores ftp.
- ***telnet_decode,*** protocolo y servicios de telnet.
- ***rpc_decode,*** protocolo RPC.
- ***frag2,*** fragmentación de paquetes.
- ***stream4 ó stream5,*** capaces de seguir 64000 conexiones simultáneas establecidas.
- ***portscan y portscan2,*** para detectar escaneos y detección de sistemas operativos.

Quizás los "*más difíciles*" de entender ahora sean ***frag2 y stream***, hablemos de ellos.

El ***preprocesador frag2*** detecta, reensambla e inspecciona paquetes fragmentados.

La fragmentación en el mundo de las redes y las comunicaciones es necesaria, los cables y medios por los que viajan nuestros datos tienen un tamaño máximo determinado (*MTU, Unidad máxima de Transmisión*) y cuando un paquete es más grande que la MTU debe fragmentarse, trocearse en "*cachitos*" más pequeños para que pueda ser enviado.

Volvemos con las analogías.... *Supongamos que nuestra "tía de Badajoz" nos quiere enviar un chorizo ibérico... bueno y un salchichón y lomo.... ahhgghhh qué hambre me está entrando*

Bien, el caso es que la empresa de transporte sólo acepta paquetes del tamaño de una caja de zapatos... entonces nuestra amada "Tía de Badajoz" trocea los embutidos, los mete en diferentes cajas, las numera, les pone las señas a cada una y las lleva al servicio de transporte.

Nadie nos garantiza que cuando recibamos tan estimados y suculentos manjares lo hagamos en el mismo orden que salió, es probable que recibamos la "caja 6/8" antes que la "caja 1/6", es más... puede ocurrir que la empresa de transporte delegue en otra el envío y que la nueva empresa utilice un MTU más bajo y "trocee" las cajas en otras más pequeñas...

Este ejemplo nos sirve para entender cómo funciona ***frag2***... los sistemas operativos, más concretamente *la pila TCP/IP del sistema operativo*, inspecciona cada paquete, es más,

algunos *routers* o *dispositivos de conmutación* puede enviar los fragmentos "*inmediatamente*" o esperar a recomponer todos... esto parece normal, pero.... *¿Y si "La Tía de Badajoz" hace una "gracia" y nos envía el chorizo ibérico en paquetes del tamaño de una caja de cerillas?*

Pues tendremos miles de trocitos, y ya sabes... es más fácil *"desordenar" que "reordenar", siempre pongo el ejemplo del puzzle, aun no conozco a nadie que sea capaz de montar un puzzle de 1.000 fichas en el mismo tiempo que yo "mezclo" esas fichas...*

Valeeee, ¿y esto por qué? Pues porque una táctica para *"mantener"* ocupados a los servidores, *routers*, etc.. consiste en **fragmentar EXCESIVAMENTE** los paquetes, primero tendrán que esperar a que los reciban todos, luego recomponer el mensaje y por ultimo ofrecer la respuesta adecuada, ***esto significa, más memoria, más tiempo y más trabajo***

Esta misma táctica se utiliza para intentar provocar un ataque por denegación de servicios a un *IDS*, a un *Firewall*, etc... de esto es de lo que se preocupa ***frag2***, de recomponer los paquetes ANTES de que sean enviados al sistema de detección, hablaremos de ello más profundamente a medida que avancemos, incluso probaremos nuestros IDS y nuestros *Firewalls* con aplicaciones específicas... *¿qué te parecería si te enviasen 12.000 correos diarios "a trozos"?* pues eso es lo que haremos, enviar miles de paquetes por segundo muy, muy, muy fragmentados...

stream4 o stream5 (la última revisión de snort) se ocupa de *realizar "un seguimiento a cada conexión"*, es decir, de monitorizar las conversaciones *"punto a punto"*, por ejemplo.... imagina que se detecta un ataque a nuestro Servidor WEB (generalmente puerto 80) por el archiconocido ***bug del code-decode***... además de *"avisarnos"* puede "*seguir atento*" a todas las acciones que vengan de la IP "*atacante*", incluso podemos configurarlo para decirle *"Mira a esta IP durante 1 hora" o "mira lo que hace está IP en los próximos 250 paquetes"....*

De hecho ***stream4/5*** puede seguir simultáneamente más de una conexión, no sólo el puerto 80 como en el ejemplo anterior, puede seguir hasta prácticamente la totalidad de servicios y puertos disponibles, el único inconveniente de ello es el trabajo excesivo, duro y difícil que tendrá nuestro PC... para esos casos precisaremos de un buen equipo, rápido y con suficientes recursos.

El sistema de detección

El ***sistema de detección*** toma los paquetes del *preprocesador* y/o de los *plug-ins* instalados y los comprueba con las reglas establecidas para los mismos, dependiendo de las acciones a realizar los dejará pasar, generará alertas, registros o incluso detendrá un eventual ataque.

La mayor parte del trabajo y optimización de ***snort*** se realiza aquí... utiliza funciones y reglas agrupadas por categorías (Troyanos, desbordamiento de buffer, acceso a aplicaciones, etc..)

El corazón del sistema de detección son las reglas, estas las impone el administrador y podemos dividirlas en dos partes:

- **Reglas de cabeceras**: Basadas en la detección de paquetes por tipo (TCP, UDP, ICMP....) o por dirección (Origen, destino, puerto)

- **Reglas de Contenido:** Son reglas que examinan el interior del paquete, es decir, el contenido de los datos transportados por el protocolo.

- **Reglas de Control:** Controlan desde la dirección del tráfico en la red hasta las contramedidas en tiempo real, pasando por la monitorización de sesiones, desbordamiento de buffers, shellcodes, negación de servicios, etc...

Siguiendo el ejemplo del cajero automático, las reglas de cabecera determinarían si se trata de un billete, de una moneda o de fraude, mientras que las reglas de contenido, descubrirán si la moneda es de 1 euro, de 2 euros, etc...

El sistema de Salida. Registro y alertas.

Una vez que se inspeccionaron las reglas y se validó el paquete algo hay que hacer con los resultados, de eso se encarga el ***sistema de salida.***

Las alertas pueden enviarse a un ***fichero de logs,*** a un equipo remoto mediante sockets o ***mensajes emergentes***, ***traps SNMP*** e incluso se pueden almacenar en ***una Base de datos SQL***

El ***sistema de salida*** también dispone de *plug-ins* que simplifican la presentación de las alertas en formatos *XML, HTML, interfaces Web y muchos otros*.

Al igual que *los plug-ins del preprocesador* será imposible detallar o describir todos, cada día los desarrolladores de software crean nuevos plug-ins, mejoras y nuevas visualizaciones, veremos unos cuantos... algunos muy interesantes.

Limitaciones, Bugs y actualizaciones

Como todo software ***snort*** es vulnerable y no siempre actúa eficientemente, en algunas ocasiones se debe a limitaciones en el hardware, como puede ser la velocidad de la tarjeta de red, agujeros de seguridad que permiten *"saltarse"* el IDS o simplemente la generación de falsas alarmas, bien por una mala elección de las reglas o por determinados paquetes generados en la red.

Entre los **fallos más comunes** al usar ***snort*** están:

- **Pérdida de paquetes**, como ya se ha dicho por problemas en la conectividad o deficiencias en las implementaciones de la pila TCP/IP del sistema operativo

- **Falsas alarmas**, pueden ser **falsos positivos** o **falsos negativos**.

Un falso positivo es una falsa alarma, un paquete o conjunto de paquetes que disparan una alerta cuando realmente son inofensivos.

Los falsos negativos son paquetes que comprometen la estabilidad de ***snort*** y que no es capaz de detectar, pueden ser debidos a errores de implementación en las reglas o a ataques contra el propio ***snort*** por vulnerabilidades descubiertas

Snort es una ***GRAN utilidad***, no exenta de ataques, pero muy, muy eficiente. No obstante y como ya estamos acostumbrados a que nada es para siempre hay que estar al tanto de actualizaciones y mejoras, las revisiones, los *updates*, los parches de seguridad.

Y no siempre las actualizaciones se deben a problemas de seguridad, pueden ser debidas a la aparición de nuevas reglas, a cambios de sintaxis... en definitiva a mejorar el programa, recuerda, la seguridad también consiste en mantenerse al día en la aparición de ***bugs*** y parches para solventar los mismos...

Instalando Snort

Ya hemos comentado la versatilidad de ***snort***, tanto por sus análisis como por la variedad de Sistemas Operativos en los que puede ejecutarse...

En esta sección nos vamos a centrar en la instalación de ***snort*** para plataformas *LiNUX* Red Hat y para Win32, concretamente para *Windows* 2000/XP

Para otras distribuciones de *LiNUX* será *"parecido"*, aunque aquí lo instalaremos mediante *RPM*. Esto significa que las *distros* de Red Hat, SuSe, Mandrake, Conectiva, etc serán aptas para seguir estos pasos.

Si usas Debian Gentoo, Slackware u otros deberás consultar las diferentes formas de instalación, mediante *apt-get install/remove, emerge o installpkg/removepkg/upgradepkg*

Puedes pasarte por los foros de ***hackxcrack*** si tienes dudas o no sabes como hacerlo... seguro que lo dominas mejor que yo :P para los que me conocéis *"soy un chico Windows"* en su 90% de trabajo y creedme que me ha costado lo mío ponerme al día con *LiNUX*... mereció la pena de verdad.

Si tienes dudas no lo dudes: *http://www.hackxcrack.com/phpBB2/index.php* *Mejor aún, actualmente estamos desarrollando un proyecto, una LIVE CD, una distribución "especial" de LiNUX personalizada, orientada a Seguridad y en la que muchos de los foreros de hackxcrack estamos colaborando... esta distribución puede ser tuya también, en ella ya figura snort como IDS y otras muchas utilidades, paquetes de seguridad, una forma fácil y sencilla de seguir estos artículos es participar en el Proyecto de la Live... es libre, todos aportamos lo que podemos, ¿podrás unirte? Te esperamos y te necesitamos.*

Volviendo al proceso de instalación....

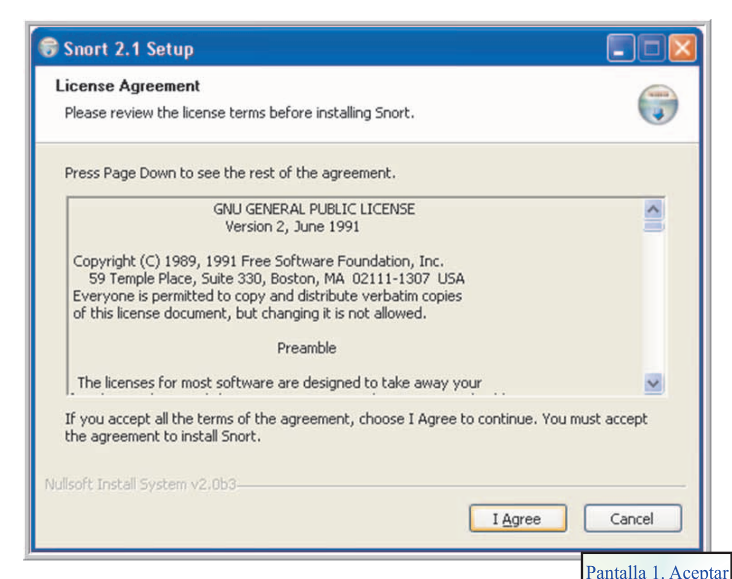
También necesitaremos instalar ***libpcap*** y si debemos compilar el código fuente necesitaremos **gcc, automake, autoconf, binutils, make...** no te asustes... lo haremos *"paso a paso"* ;)

La instalación de ***snort*** en plataformas *Windows* sigue las *"directrices de siempre" Siguiente-Siguiente-Siguiente...* y en este caso, en lugar de las librerías *libpcap* se necesitará ***WinPcap***.

Instalando snort en *Windows* 2000/XP

Lo primero bajarse el *IDS* http://www.snort.org/dl/binaries/win32/ una vez en nuestro PC se instala como la práctica totalidad de aplicaciones *Windows*....

Paso 1) Aceptar los Términos de Licencia, recuerda que ***snort* es un producto GNU**
Paso 2) Seleccionar la configuración.



Pantalla 1. Aceptar los términos de la licencia

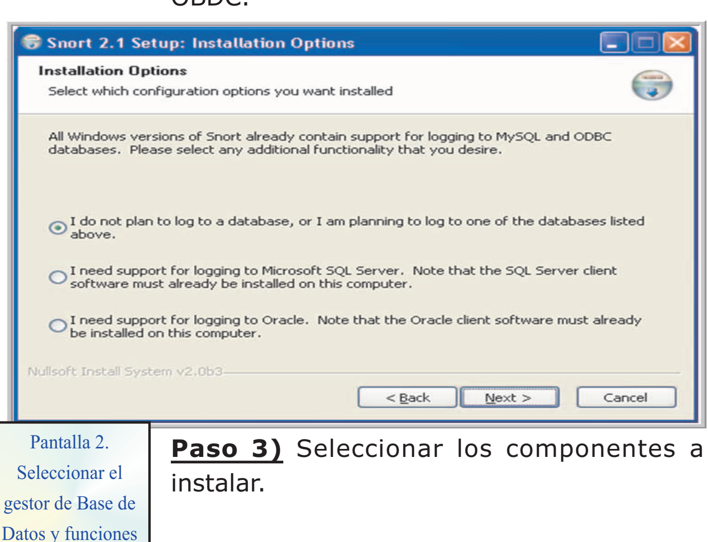
Quizás sea el único paso que merece la pena comentar, todas las implementaciones de ***snort*** para sistemas Win32 incluyen el soporte para Bases de Datos MySQL y ODBC, si deseamos mantener los registros y alertas de ***snort*** en otras Gestores de Bases de Datos como SQL Server u ORACLE seleccionaremos las opciones 2 ó 3 respectivamente, no será nuestro caso, nosotros, cuando llegue la ocasión, usaremos MySQL tanto para plataformas *LiNUX* o *Windows*.

Tal y como indica la pantalla, si deseamos usar SQL Server u ORACLE, se debe disponer tanto de los Gestores como de los clientes necesarios para ellos.

¿Por qué MySQL? Muchas podrían ser las respuestas, principalmente por:

- En números anteriores de la revista ya se enseñó a instalar y configurar MySQL
- MySQL es muy eficiente y además es Libre
- No dependeremos de un Sistema Operativo en concreto
- Muchos *plug-ins, scripts* y programas de terceros se brindan con soporte MySQL

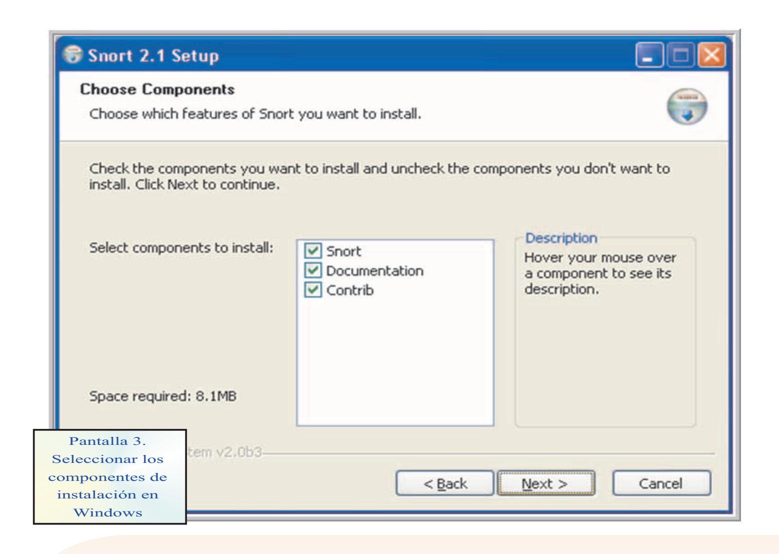
Por tanto en la pantalla que sigue escogemos la primera opción la cual nos posibilitará almacenar logs en Bases de Datos MySQL u OBDC.



Pantalla 2. Seleccionar el gestor de Base de Datos y funciones adicionales

Paso 3) Seleccionar los componentes a instalar.

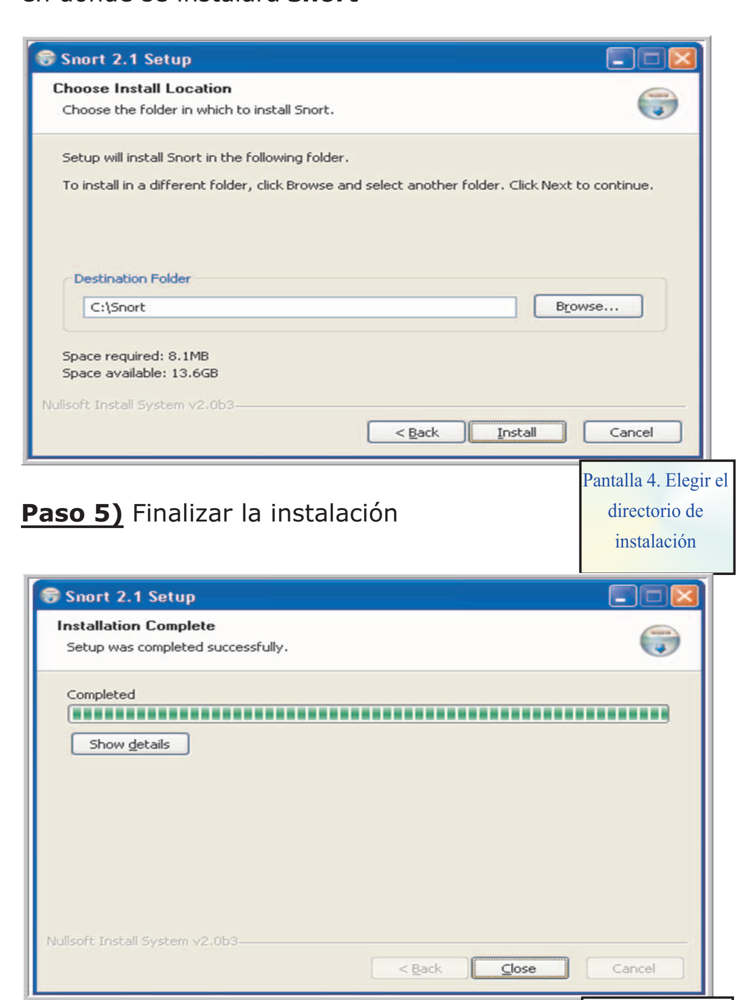
Verificamos todos los que aparecen si no lo están ya, ***snort***, la ***documentación*** y ***Contrib*** (que contiene *plug-ins* y añadidos que *contribuyen* al manejo de ***snort***)



Pantalla 3. Seleccionar los componentes de instalación en Windows

Paso 4) Directorio de Instalación.

Elegimos C:\snort como carpeta o directorio en donde se instalará ***snort***



Pantalla 4. Elegir el directorio de instalación

Paso 5) Finalizar la instalación

Pantalla 5. Finalizar la instalación

Una vez finalizada la instalación debemos instalar las librerías ***pcap***, en *Windows*, ***WinPcap***

Sólo algunas salvedades.... Las librerías Pcap (***ya sean libpcap o WinPcap***) operan en la ***capa 2 del modelo OSI (enlace a datos)*** y son necesarias para que ***snort*** capture el tráfico que circula por la red.

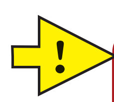
En el caso particular de *Windows*, puedes descargar la última versión de ***WinPcap*** en

http://winpcap.polito.it/

Actualmente está en desarrollo la ***versión 3.1 beta de WinPcap***, **no te recomiendo** que instales la última, a mi me dio bastantes problemas, sobre todo si usamos ***snort*** directamente sobre equipos con MODEM analógico y protocolos *ppp*.

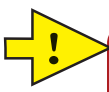
Con la versión 2.3 funciona perfectamente, si no encuentras en la web oficial (http://winpcap.polito.it/) el link de descarga de ***WinPcap 2.3*** puedes bajarlo directamente de nuestro Servidor Virtual.

http://www.forohxc.com/ids/snort/winpcap/2.3/WinPcap_2_3.exe



Los programas...

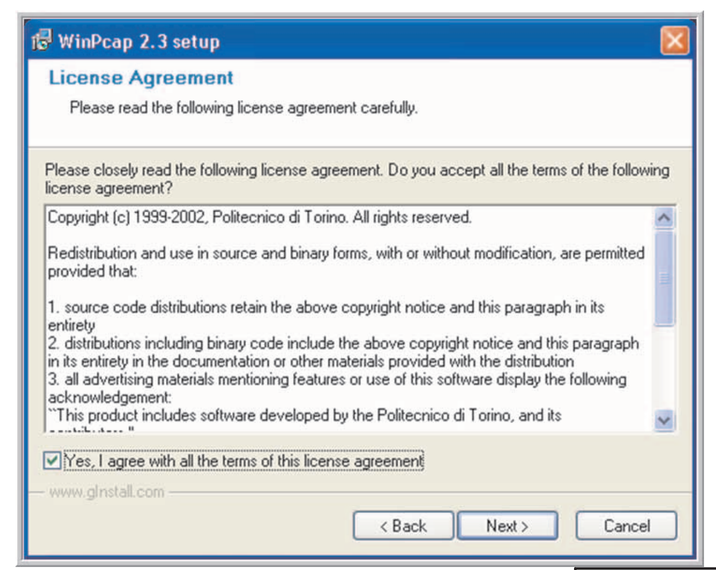
Los programas utilizados en el presente artículo también pueden ser descargados desde la Web de la revista PC PASO A PASO: www.hackxcrack.com (en la sección "Artículos Liberados y Descargas").



IMPORTANTE

IMPORTANTE: Antes de instalar las librerías comprueba que no tengas otras versiones de la misma instaladas en tu PC, muchos esnifers y productos de Windows utilizan WinPcap y hasta es posible que ya dispongas de esta u otras versiones sin conocerlo.... no creo que haga falta decirte a estas alturas como se desinstala un software en Windows no? Bueeeeeno, Inicio-Configuración-Panel de Control-Agregar o quitar programas.... busca en la lista si aparece WinPcap y la desinstalas... después instalas la que te acabas de descargar.

La instalación de ***WinPcaP*** no tiene ningún misterio, es totalmente automatizada y lo único que debemos hacer es ir pulsando en el botón de ***Next, Next, Next*** a medida que nos van apareciendo pantallas (y aceptar los términos de la licencia cuando aparezca, como en la pantalla que se muestra a continuación)



Pantalla 6. Aceptar los términos de la licencia de WinPcap

Aunque no lo dice, tras finalizar la instalación mejor reiniciar el equipo....

Instalando snort en *LiNUX* desde RPM

Como antes, lo primero el lugar donde encontrarlo:
http://www.snort.org/dl/binaries/*LiNUX*/

Entre todos ellos

1º) http://www.snort.org/dl/binaries/linux/snort-2.1.1-1.i386.rpm

2º) http://www.snort.org/dl/binaries/*LiNUX*/snort-mysql-2.1.1-1.i386.rpm

Tras descargar ***snort*** (los dos enlaces anteriores) y alojarlo en el directorio elegido podremos instalarlo bien **desde consola o** bien desde ***X Window***

1.- Desde consola:

rpm –Uvh snort-mysql-2.1.1-1.i386.rpm si lo que deseamos es **actualizar** una versión ya existente, o bien,

rpm –i snort-mysql-2.1.1-1.i386.rpm para **instalar *snort* por primera vez**. Advierto que si ya dispones de una versión instalada

y configurada con reglas personalizadas, etc.. utilizar este método puede significar perderlo todo y empezar de cero.

2.- Desde XWindow

Usando escritorios gráficos como KDE, GNOME u otros, simplemente hacer doble clic en el archivo ***RPM*** descargado

Instalar las librerías PCAP

Al igual que las versiones *Windows*, debemos de tener las ***librerías pcap***, aquí son ***libpcap*** y no *WinPcap*, para que ***snort*** funcione correctamente.

Red Hat y otras distribuciones de *LiNUX* ya las incluyen en la instalación, si no es tu caso o si quieres actualizar la revisión de las librerías puedes seguir los pasos que vienen a continuación, en caso contrario tu ***snort*** ya está preparado.

Las últimas revisiones de ***libpcap*** las puedes encontrar en http://www.tcpdump.org la última revisión en el momento de escribir este artículo es la 0.8.1 y la puedes descargar directamente de aquí: http://www.tcpdump.org/release/libpcap-0.8.1.tar.gz

Para instalar ***libpcap*** desde los archivos fuente basta con compilar el código, el único requisito es elegir de las utilidades de desarrollo (***development tools***) en la distribución que uses.

En el ejemplo que sigue a continuación describe los pasos para instalar ***libpcap en Red Hat 8.***

Necesitaremos:

gcc, El compilador de C

automake, utilidad para crear los makefiles "al vuelo"

autoconf, utilidad para configurar el código fuente "al vuelo"

binutils, utilidades para binarios

make, utilidad para compilar código individual

Paso 1) Selección de paquetes de instalación

Desde el usuario ***root***, seleccionar ***Configuración del sistema – Paquetes***

Y se abrirá el cuadro de diálogo Add or Remove Packages

Paso 2) Mover la barra de desplazamiento hasta la sección ***Development*** y verificar la casilla de verificación ***Herramientas de desarrollo***



Pantalla 7. Añadir o quitar paquetes y herramientas de desarrollo (Development)

Paso 3) Pinchar en ***Actualización***

El sistema operativo calculará las dependencias y paquetes necesarios.

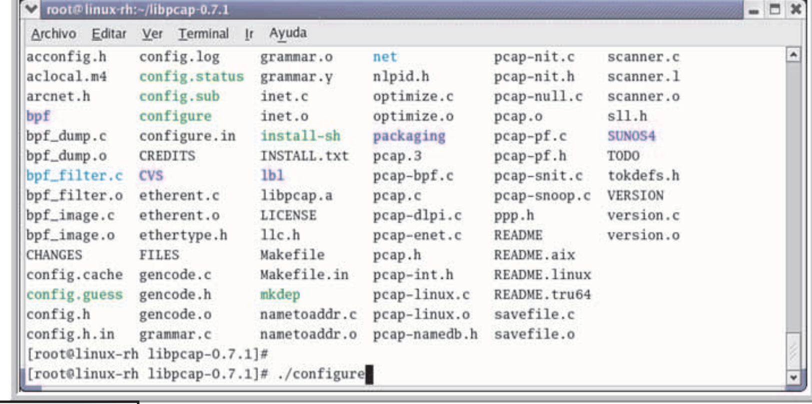
Paso 4) Si se desea pinchar en ***Mostrar detalles*** para ver las opciones seleccionadas

Paso 5) Pinchar en ***Continuar*** y el sistema procederá a la instalación
Ahora que el sistema ya tiene los paquetes necesarios para ya puedes usar ***configure, make y make install*** para instalar la librerías,

para ello descomprime el archivo ***libpcap-0.8.1.tar.gz,*** abre una sesión de consola y sigue estos pasos:

1º) Ejecutar el script ./configure

./configure es un *script* que contiene la información necesaria de la máquina en la que está corriendo, las dependencias, variables y verifica el software.



Pantalla 8. Listado de archivos fuente de las librarías libpcap para plataformas LiNUX

También es el responsable de generar los *makefiles* necesarios y en caso que se produzcan errores nos avisará de ello.

2º) Compilar el código fuente con make

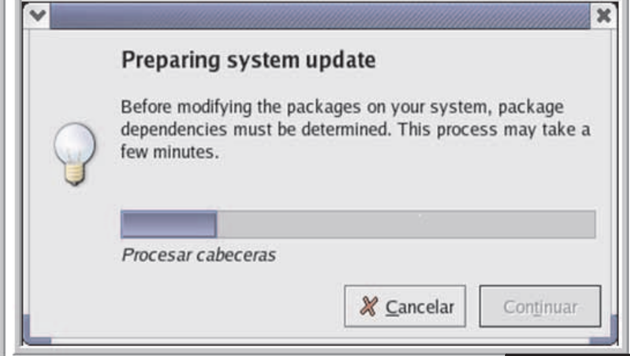
make es un comando que existe en la práctica totalidad de distribuciones *LiNUX*, no es un *script* como *configure*, ***make*** usará el *makefile* creado por *configure* y su función es compilar el código que será usado durante la instalación final, es como un archivo de inicialización y al igual que antes en muy importante que no se hayan producido errores durante el proceso.

3º) Instalar las librerías usando make install

make install, esto es el paso final del proceso. Lee la información del *makefile* creado, la distribución que está siendo usada y los ficheros ejecutables u otros archivos creados en la estructura de directorios. Tras finalizar este paso sin errores, el software estará instalado y disponible para su uso. Otra forma es encontrar un ***RPM*** listo para instalarse en la distribución que usemos... puedes encontrarlas aquí:

http://www.rpmfind.net/*LiNUX*/rpm2html/search.php?query=libpcap

Y puedes seguir los mismos pasos indicados para instalar ***snort*** desde ***RPM*** para *libpcap*.



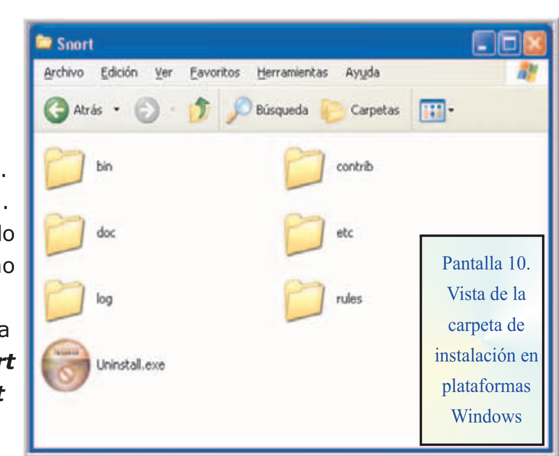
Pantalla 9. Instalación desde una distribución RPM

Bueno, también te puede dar por compilar y crear los ***configure-make-make install*** desde el código fuente de ***snort***, eso lo dejo a tu elección... si eres un usuario poco avezado en *LiNUX* como lo soy yo... será una buena práctica personal y te ayudará a resolver otras muchas situaciones cuando se tenga que compilar otros códigos fuente, en el próximo artículo esta acción será de uso obligado puesto que no siempre los binarios se compilaron con todas las opciones disponibles ;)

Verificando la instalación de snort

Plataformas *Windows*

No busques ***snort*** en agregar o quitar programas... no está allí... si has seguido los pasos uno por uno lo tendrás en la carpeta ***snort*** en ***C:\snort***



Pantalla 10. Vista de la carpeta de instalación en plataformas Windows

Esta no es la forma de probar ***snort***, pero al menos estamos viendo las carpetas que se crearon durante el proceso de instalación y un programa unistall.exe para eliminarlo.

bin están los binarios y programas para ejecutar ***snort***

doc, la documentación y ayudas

log, se guardaran los registros de las intrusiones

Contrib, encontrarás programas y archivos que contribuyen o mejoran el funcionamiento

etc, hay algunos archivos para mantener alertas, reglas, pero sobre todo un archivo muy, muy especial... ***snort.conf*** que es el archivo de configuración del programa, algo así como el lugar donde ***snort*** acudirá para saber en el modo que ha de ejecutarse, las reglas que debe incluir, variables, directivas del *pre-procesador*... etc...

rules, se guardan las reglas que se podrán aplicar.

Plataformas *LiNUX*

En *LiNUX* los lugares en donde se instala ***snort*** es diferente...

/usr/bin/snort están los binarios y programas para ejecutar ***snort***

/usr/share/man/man8/snort.8.gz, la documentación y ayudas

/var/log/snort/, se guardaran los registros de las intrusiones

/**etc/snort**, está el archivo de configuración y otros

/etc/snort/rules/ Las reglas

¿Y qué fue de ***Contrib****?*

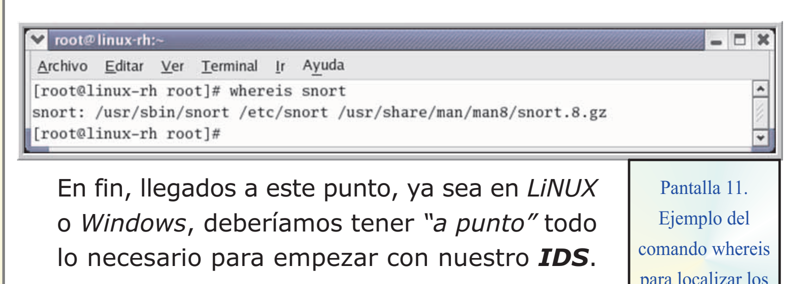
Pues en *LiNUX* no se instala a menos que descarguemos las contribuciones de terceros:

http://www.snort.org/dl/contrib/

También puedes actualizar las reglas en:

http://www.snort.org/dl/rules/

También puedes usar ***whereis*** para encontrar los lugares donde está repartido ***snort***



Pantalla 11. Ejemplo del comando whereis para localizar los binarios de snort en LiNUX

En fin, llegados a este punto, ya sea en *LiNUX* o *Windows*, deberíamos tener *"a punto"* todo lo necesario para empezar con nuestro ***IDS***.

Aunque para este primer artículo no es preciso disponer de las aplicaciones que vienen a continuación, lo serán más adelante, así que, vete instalando si los tienes aún las siguientes:

- ***Un Servidor Web***, mejor Apache aunque uses *Windows*... aunque también puede ser ISS
- El gestor de Bases de Datos, ***MySQL*** (imprescindible si instalamos ***snort*** con soporte MySQL)
- ***Intérprete de perl*** para *Windows* o *LiNUX*
- ***Lenguaje PHP*** para la versión de tu sistema operativo.

Todos estos programas y aplicaciones ya han fueron usadas en artículos anteriores en la revista, revisa los números donde se hablaba de ***dsniff*** (en donde se usaba algún que otro script en *perl*) y en el artículo de **creación de foros *phpBB2* (Apache+MySQL+PHP)**

Si no dispones de los programas los encontrarás en:

http://httpd.apache.org/download.cgi

http://www.php.net/downloads.php

http://www.mysql.com/downloads/index.html

http://www.perl.com/pub/a/language/info/software.html

El archivo de Configuración de snort: Snort.conf

Si editamos el archivo ***snort.conf*** veremos algo así:

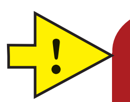
```
snort.conf - Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
#  4) Customize your rule set
#
####################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
# $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:

var HOME_NET any

# Set up the external network addresses as well.  A good start may be "any"
var EXTERNAL_NET any

# Configure your server lists.  This allows snort to only look for attacks to
```

Pantalla 12. Contenido del archivo de configuración. Snort.conf



Donde está...

¿Dónde está el archivo **snort.conf**?

Venga, vale… En Windows hemos instalado el **SNORT** en **c:\snort** ¿verdad?, pues el archivo **snort.conf** está en la carpeta **c:\snort\etc**

¿cómo podemos abrir y editar el archivo snort.conf? No, si picas con el Mouse dos veces sobre el archivo NO SE ABRE, claro, puesto que tiene extensión **.conf** y nuestro Windows "pasa" de él. Tienes que abrirlo con cualquier editor de texto plano, por ejemplo el **Bloc de Notas** de nuestro Windows :)

Y para los que crean que no es necesario explicar algo tan básico, bufff, tendrían que ver los correos electrónicos que nos llegan a la editorial ;)

Es un archivo en texto plano que indica a ***snort*** cómo debe actuar, las redes de que disponemos, las reglas, las variables *"de entorno"*, las directivas del *pre-procesador* , las alertas...

No es el momento ahora de hablar profundamente del mismo, pero es importante que modifiques o configures el archivo a las características de tu red y preferencias...

Snort usa variables para conocer la red, máquinas, etc... por omisión los valores son:

Variable name	IP Address/Range
HOME_NET	any
EXTERNAL_NET	any
DNS_SERVERS	$HOME_NET
SMTP_SERVERS	$HOME_NET
HTTP_SERVERS	$HOME_NET
SQL_SERVERS	$HOME_NET
TELNET_SERVERS	$HOME_NET
SNMP_SERVERS	$HOME_NET
HTTP_PORTS	80
SHELLCODE_PORTS	!80
ORACLE_PORTS	1521
AIM_SERVERS	[64.12.24.0/24,64.12.25.0/24,64.12.26.14/...
RULE_PATH	../rules

Tabla . Principales variables definidas por snort

Es imprescindible que ahora modifiques las variables con los valores correspondientes al entorno que usas, para las prácticas que vienen a continuación usaremos este escenario:

<u>Red interna:</u> 172.28.0.0 /16
<u>Red externa:</u> cualquier dirección IP, red o subred
<u>Servidor web:</u> 172.28.0.99

Al menos tienes que modificar las ***variables HOME_NET, EXTERNAL_NET, HTTP_SERVERS. HTTP_PORTS y RULE_PATH***, para ello busca dentro del archivo ***snort.conf*** las declaraciones de estas cuatro variables y dales los valores que necesiten, obviamente es posible que no manejes los mismos rangos de red/subred que los míos, adáptalos a tu configuración personal

var HOME_NET 172.28.0.0/16

var EXTERNAL_NET any

var HTTP_SERVERS 172.28.0.24

var HTTP_PORTS 80

var RULE_PATH ../rules (en *Windows*)

var RULE_PATH /etc/snort/rules (en *LiNUX*)

Vamos a probarlo, escribe **snort –W** en la ventana de comandos y pulsa enter, pongo la captura de *Window**s.***

```
C:\WINDOWS\System32\cmd.exe
C:\Snort\bin>snort -W

-*> Snort! <*-
Version 2.1.1-ODBC-MySQL-FlexRESP-WIN32 (Build 24)
By Martin Roesch <roesch@sourcefire.com, www.snort.org>
1.7-WIN32 Port By Michael Davis <mike@datanerds.net, www.datanerds.net/~mike>
1.8 - 2.1 WIN32 Port By Chris Reid <chris.reid@codecraftconsultants.com>

Interface       Device          Description
-------------------------------------------
1  \Device\Packet_{81F249AF-04B0-4384-A31E-00DF6B93D431} (Realtek RTL8139/810x F
amily Fast Ethernet NIC                    )
2 \Device\Packet_NdisWanIp (NdisWan Adapter)
3 \Device\Packet_NdisWanBh (NdisWan Adapter)


C:\Snort\bin>
```

Pantalla 14. Mostrar los interfaces de red disponibles en Windows con -W

En este caso hay 3, dos para WAN y 1 para *ethernet*... este equipo es un portátil y dispone de un MODEM interno que también puede ser usado para monitorizar el tráfico con ***snort***.

Seleccionar la tarjeta de red en ambas plataformas se realiza con la opción **–i** sequida del número de tarjeta de red deseada (*Windows*) o **–i** tarjeta de red (*LiNUX*, p.e. **–i eth0**), tranquilo, ahora lo veremos en la practica 2.

Tampoco puedo ahora extenderme mucho en cada una de las opciones, ya sabes... problemas de espacio del artículo, por el momento veamos estas:

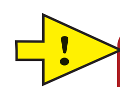
-D, ejecuta ***snort*** como demonio o servicio
-a, habilita la decodificación de protocolo *ARP*
-e, similar a la anterior
-C decodifica los resultados en caracteres *ASCII*
-v, muestra la decodificación detallada
-d, volcado de los paquetes capturados
-l, indica un directorio donde se guardarán los *logs*
-p, inicializa ***snort*** en modo no promiscuo
-N, no guarda *logs*
-n, indica que tras n paquetes recibidos se tome una acción, por ejemplo terminar ***snort***
-y, muestra el año junto con la decodificación o *logs*
-U, fecha/hora del análisis
-c indica el directorio del archivo de configuración y sus reglas (***snort.conf***)

Hay mas.... MUCHAS MAS... las iremos viendo a medida que avancemos, y no te equivoques en el uso entre mayúsculas y minúsculas, no es lo mismo –A que -a

Práctica 2. Ejecutar snort como un esnifer

La forma más simple de ejecutar ***snort*** como un ***esnifer*** de paquetes usando algunas de las opciones mencionadas es:

snort –dev (y pulsamos enter)



IMPORTANTE

IMPORTANTE:

Antes hemos dicho que mediante la instrucción **snort –W** podríamos visualizar las “tarjetas de red” que tenemos, en nuestro caso han sido llamadas por Windows **1**, **2** y **3** (pantalla 14).

Ahora te acabamos de decir que ejecutes la orden **snort –dev,** bien, en este caso “snifaremos” por defecto la tarjeta de red **1.**

Pero en tu caso particular, puedes tener una o dos o tres o 200 “tarjetas de red” (cada uno tiene el equipo que puede comprar). Imagina que en tu caso particular la tarjeta que está conectada a Internet es la **2** y queremos “snifar” precisamente esa. En tal caso, en lugar de escribir **snort –dev** deberemos escribir **snort –i 2 –dev** (y pulsar enter).

-i 2 le dice al **snort** que “snife” la tarjeta de red número 2

y podremos ver cosas como estas...

```
C:\WINDOWS\System32\cmd.exe - snort -dev

-*> Snort! <*-
Version 2.1.1-ODBC-MySQL-FlexRESP-WIN32 (Build 24)
By Martin Roesch <roesch@sourcefire.com, www.snort.org>
1.7-WIN32 Port By Michael Davis <mike@datanerds.net, www.datanerds.net/~mike>
1.8 - 2.1 WIN32 Port By Chris Reid <chris.reid@codecraftconsultants.com>
03/21-14:25:52.173470 ARP reply 172.28.0.1 is-at 0:C0:49:D4:5F:CD

03/21-14:25:52.173508 0:0:39:C1:6A:C2 -> 0:C0:49:D4:5F:CD type:0x800 len:0x55
172.28.0.50:1029 -> 195.235.113.3:53 UDP TTL:128 TOS:0x0 ID:466 IpLen:20 DgmLen:
71
Len: 43
00 0A 01 00 00 01 00 00 00 00 00 00 03 32 30 30  .............200
01 30 02 32 38 03 31 37 32 07 69 6E 2D 61 64 64  .0.28.172.in-add
72 04 61 72 70 61 00 00 0C 00 01                 r.arpa.....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Pantalla 15.Ejemplo de salida estándar de snort

cuando pulsemos ***CTRL.+c*** se detendrá ***snort***... y mostrará las estadísticas:

De hecho no hubiese sido preciso modificar nada, si lo dejamos como está funcionarán igual de bien nuestras prácticas pero es mejor *"delimitar"* el alcance de *snort* y empezar a manipular el archivo... más adelante será necesario.

Si te estás preguntando si se pueden indicar más de una red interna o externa, más de un puerto para el *web server*, etc... la respuesta es SI, sería así:

```
var INTERNAL_NETS [172.28.0.0/16,192.168.10.0/24,10.0.0.0/8]
var HTTP_PORTS 80:85
```

Estas líneas definen tres redes internas y le dicen a ***snort*** que el *webserver* escucha peticiones por los puertos, 80, 81, 82, 83, 84, ó 85....



Buff!!!...

Buff!!!, no puedo, no tengo espacio para explicar lo que son las redes, subredes, máscaras de subred, CIDR.... En los foros de hackxcrack (www.hackxcrack.com) podremos ayudarte, además de varios links ya existentes podemos explicártelo un poquito más, ahora no es el momento, por otra parte la Revista acaba de iniciar un curso de TCP/IP que seguro responde a esas dudas, también el los foros hicimos un Taller de TCP/IP hace unos meses, pásate por allí si no sabes cual es el formato CIDR que debes aplicar a tu dirección de red...

Práctica 1. Ejecutar snort y opciones de ejecución

Lo primero que haremos es ejecutar ***snort*** *"sin más"*.

En *LiNUX* podremos hacerlo simplemente escribiendo ***snort***.

En *Windows*:

- Abriremos una de nuestras famosas "ventanitas negras" (Ventana de Comandos). Ya se ha explicado mil veces -----------> Menu Inicio --> Todos los programas --> Accesorios --> Símbolo del Sistema
- Nos situaremos en el directorio de instalación (***C:\snort\bin\snort***, si seguiste los pasos indicados anteriormente). Ya sabes, escribiremos en la ventanita negra la orden **cd c:\snort\bin** (y pulsaremos enter) Ejecutaremos el programa, ya sabes, escribiremos **snort** (y pulsaremos enter)

Símbolo del sistema

```
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\RUBEN>cd c:\snort\bin\

C:\Snort\bin>snort

-*> Snort! <*-
Version 2.1.2-ODBC-MySQL-FlexRESP-WIN32 (Build 25)
By Martin Roesch (roesch@sourcefire.com, www.snort.org)
1.7-WIN32 Port By Michael Davis (mike@datanerds.net, www.datanerds.net/~mike)
1.8 - 2.1 WIN32 Port By Chris Reid (chris.reid@codecraftconsultants.com)
USAGE: snort [-options] <filter options>
       snort /SERVICE /INSTALL [-options] <filter options>
       snort /SERVICE /UNINSTALL
       snort /SERVICE /SHOW
Options:
        -A         Set alert mode: fast, full, console, or none  (alert file ale
rts only)
        -b         Log packets in tcpdump format (much faster!)
        -c <rules> Use Rules File <rules>
        -C         Print out payloads with character data only (no hex)
        -d         Dump the Application Layer
        -e         Display the second layer header info
        -E         Log alert messages to NT Eventlog. (Win32 only)
        -f         Turn off fflush() calls after binary log writes
        -F <bpf>   Read BPF filters from file <bpf>
        -h <hn>    Home network = <hn>
        -i <if>    Listen on interface <if>
        -I         Add Interface name to alert output
        -k <mode>  Checksum mode (all,noip,notcp,noudp,noicmp,none)
        -l <ld>    Log to directory <ld>
        -L <file>  Log to this tcpdump file
        -n <cnt>   Exit after receiving <cnt> packets
        -N         Turn off logging (alerts still work)
        -o         Change the rule testing order to Pass|Alert|Log
        -O         Obfuscate the logged IP addresses
        -p         Disable promiscuous mode sniffing
        -P <snap>  Set explicit snaplen of packet (default: 1514)
        -q         Quiet. Don't show banner and status report
        -r <tf>    Read and process tcpdump file <tf>
        -R <id>    Include 'id' in snort_intf<id>.pid file name
        -s         Log alert messages to syslog
        -S <n=v>   Set rules file variable n equal to value v
        -T         Test and report on the current Snort configuration
        -U         Use UTC for timestamps
        -v         Be verbose
        -V         Show version number
        -W         Lists available interfaces. (Win32 only)
        -w         Dump 802.11 management and control frames
        -X         Dump the raw packet data starting at the link layer
        -y         Include year in timestamp in the alert and log files
        -z         Set assurance mode, match on established sesions (for TCP)
        -?         Show this information
<Filter Options> are standard BPF options, as seen in TCPDump

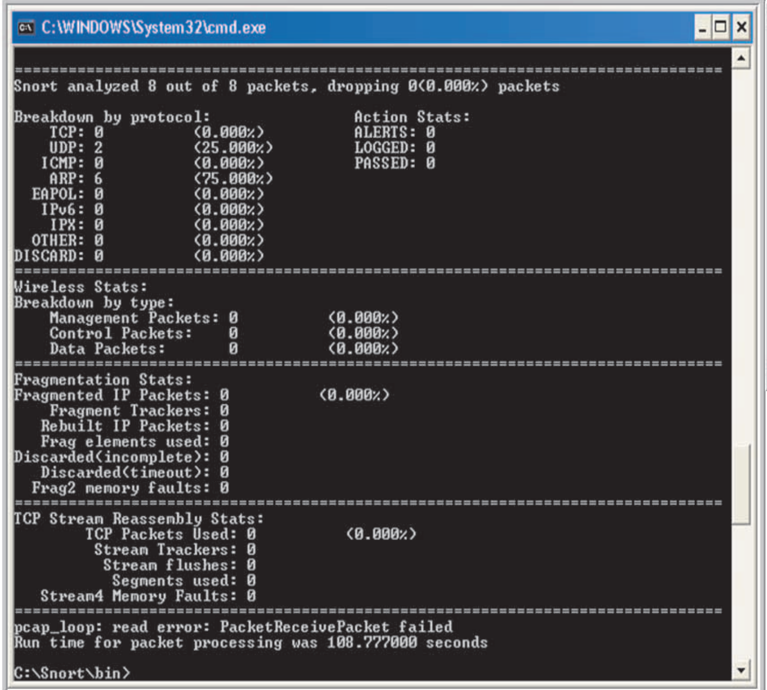

Uh, you need to tell me to do something...

: No such file or directory

C:\Snort\bin>
```

Pantalla 13. Opciones de línea de comando para ejecutar snort

Las opciones y sintaxis con las que se puede iniciar **snort** en ambas plataformas son prácticamente idénticas, alguna nueva en *Windows* como **–W** "en mayusculas" (que muestra una lista de las interfaces de red disponibles) pero poco más, si todo fue bien aparecerá algo así: (pantalla 14)



Pantalla 16. Resultado y estadísticas tras pulsar CTRL.+C

Complicado, verdad?

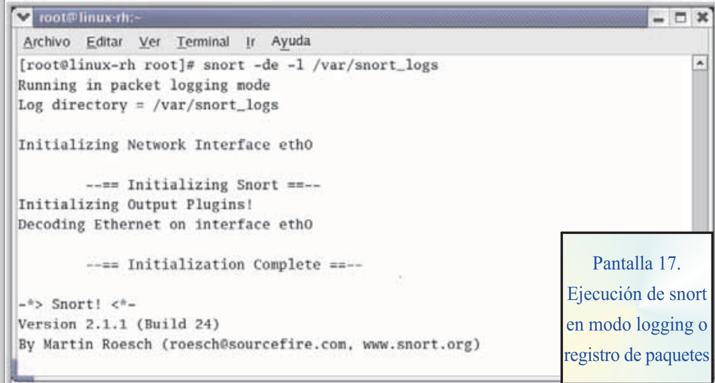
Ciertamente lo complicado es saber interpretar la captura del *esnifer*, no te preocupes por ahora si no lo entiendes, de momento lo que importa es que "*salgan cosas*", luego veremos con más detalle alguna de estas salidas.

Para los que hayáis seguido el *Taller de TCP/IP* que hicimos en los foros de *HackXcrack* no tendrá ningún misterio saber interpretar la salida del *esnifer*, los que no lo hayáis hecho tendréis dudas... bueno, no importa... en estos momentos la revista está siguiendo un curso de TCP/IP, esto te ayudará y también estos mismos artículos cuando llegue el momento.

Práctica 3. Ejecutar snort como un capturador de logs

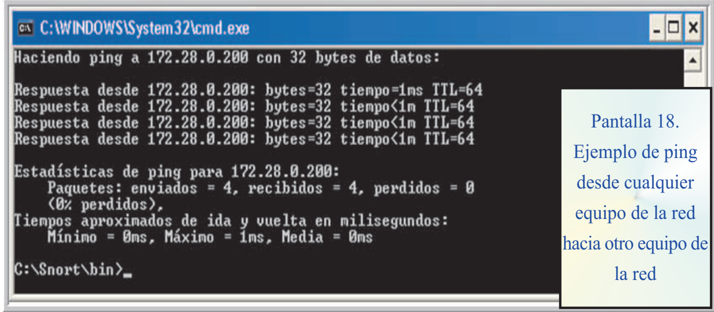
Bueno, esta la vamos a hacer desde *LiNUX*... para que no me digan luego que *"todo es Windows"*, lo único que tendremos que hacer es indicarle a ***snort*** donde debe guardar los registros y que ese directorio exista, claro...

snort –de –l /var/snort_logs



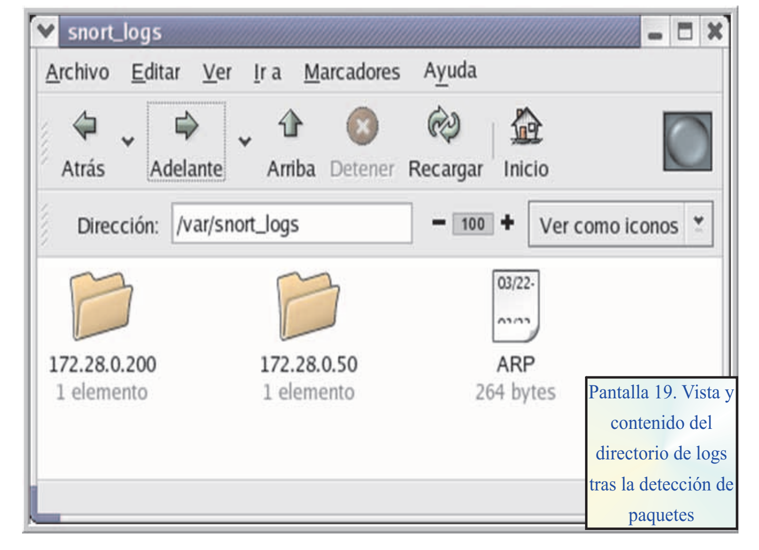
Pantalla 17. Ejecución de snort en modo logging o registro de paquetes

Ahora, desde otro equipo ***hacemos un ping a la dirección IP donde está corriendo snort***



Pantalla 18. Ejemplo de ping desde cualquier equipo de la red hacia otro equipo de la red

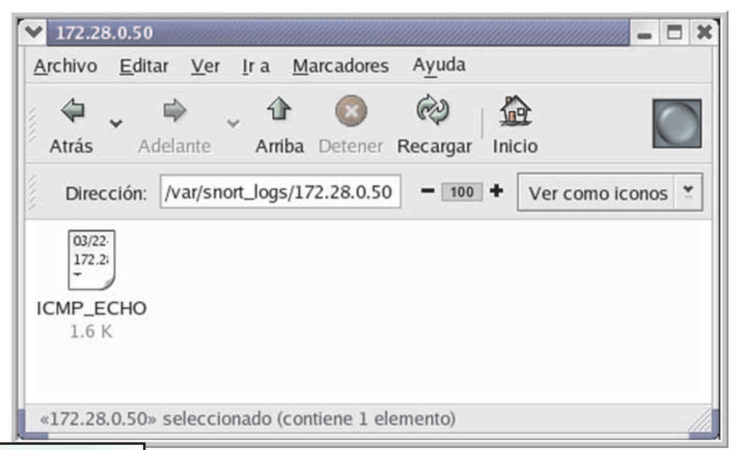
Si miramos en el directorio ***/var/snort_logs***, veremos esto:



Pantalla 19. Vista y contenido del directorio de logs tras la detección de paquetes

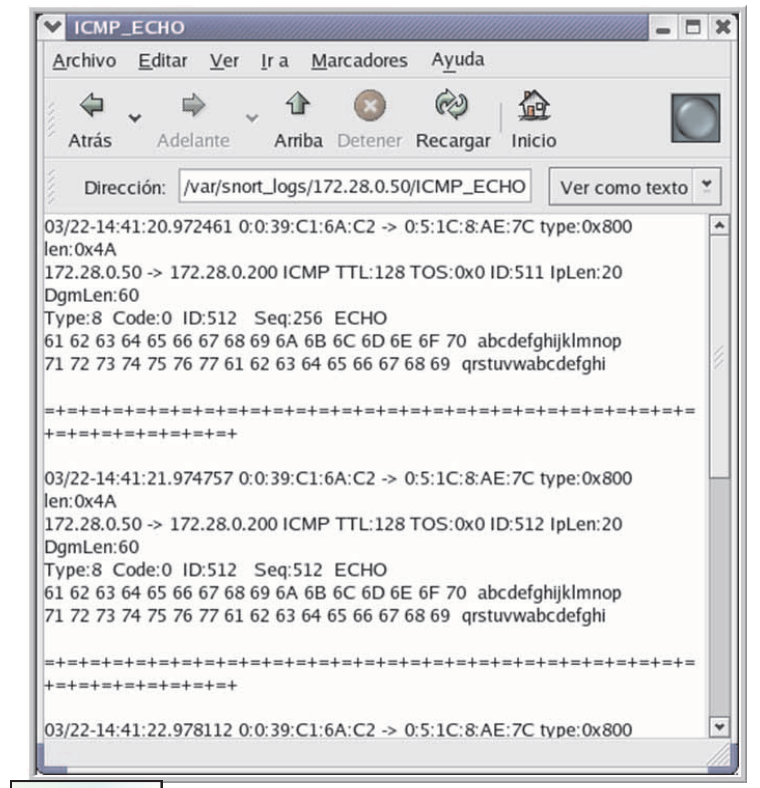
Verás que se **crearon varias carpetas**... ***snort*** generará una por cada dirección IP que registre (172.28.0.50 la del equipo que hizo el ping, etc...)

Ahora abriremos esa...



Pantalla 20. Contenido específico de una de las carpetas de registro

Hay un archivo que corresponde a la petición echo (la solicitud del ping)



Pantalla 20. Contenido específico de una de las carpetas de registro

Práctica 4. snort como un NIDS usando alertas y reglas

Bueno, me quedo sin espacio... o ya me he pasado..., **vamos a realizar una práctica global que monitorice las peticiones ping a cualquier equipo de la red, nos alerte de ello y dispare alarmas ante un posible acceso mediante el bug de unicode a nuestro servidor web,** ya sabes... *el bug de code-decode* tan "*pasado*" y ampliamente comentado en esta revista y en los foros, pero es algo a lo que "*ya estamos habituados*" y sabemos como funciona... ya verás, ya... hay muchas otras vulnerabilidades que pueden ser descubiertas gracias a ***snort*** y no me refiero sólo a las de los servidores web.... *ftp, telnet, rpc, netbios....*

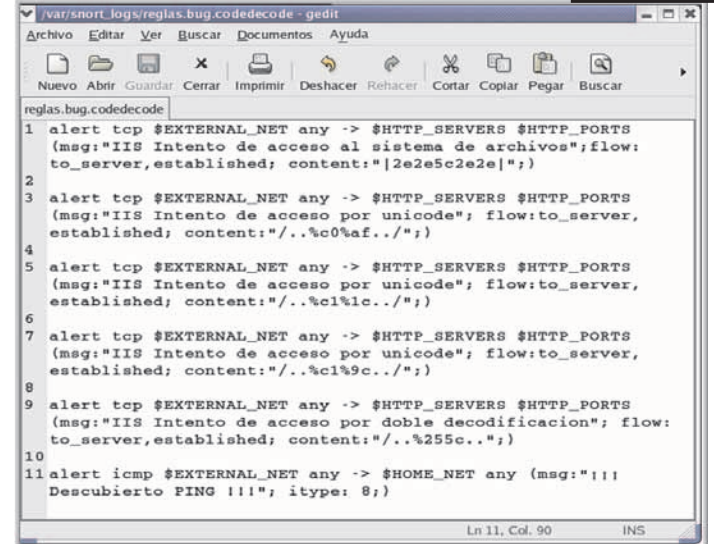
Esta práctica no la detallaré... *tendrás que tener Fé, confiar en lo que ponga y repetir lo mismo...* es una introducción a lo que nos vendrá mas tarde:

Disponemos de **un sistema *snort* corriendo en nuestra red interna 172.28.0.0/16** que vigila desde **un equipo LiNUX RH8 de ip 172.28.0.200** y **un servidor web con dirección IP 172.28.0.99**, lo que pretendemos es registrar los *pings* a cualquier equipo de la red y los intentos de explotar el *bug de unicode*.

Un dato importante... no es preciso que el *webserver* sea vulnerable... ***snort*** recogerá los intentos sea o no sea vulnerable al *"acceso transversal al sistema de archivos".* Debemos tener el archivo ***snort.conf*** bien configurado y hay que aplicar alguna regla...

Paso 1º) Crear un archivo de texto el cual **contiene las reglas** de nuestro escenario:

Pantalla 22. Definir un archivo de reglas personalizado

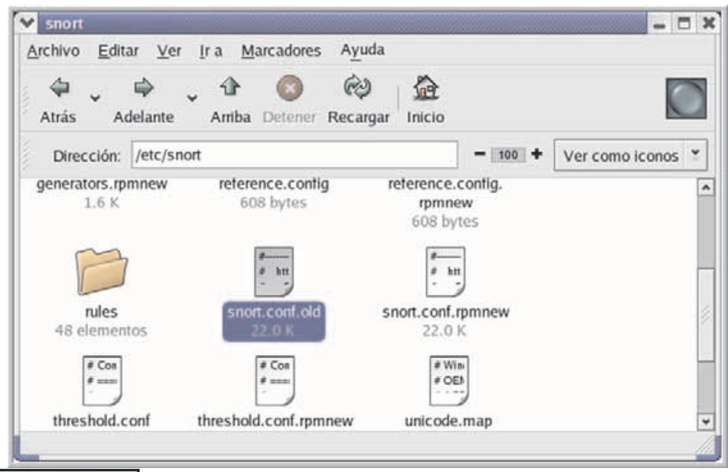


Y lo guardaremos en algún lugar de nuestro disco, elegí ***/var/snort_logs/*** puede ser otro siempre y cuando el archivo de configuración apunte a ese directorio, el nombre de ese archivo será ***reglas.bug.codedecode***, puede ser otro también... y como antes debe corresponderse con la información que demos en el ***paso 3º)***

Escribe todas esas reglas en una sola línea.

A poco que seamos algo espabiladillos entenderemos las reglas... pero no te preocupes si no las *"cazas"* ahora, esto es parte de próximos artículos...

Paso 2º) Renombraremos el archivo de configuración situado en ***/etc/snort/snort.conf como /etc/snort/snort.conf.old*** (En *Windows* está situado en C:\snort\etc\snort.conf)

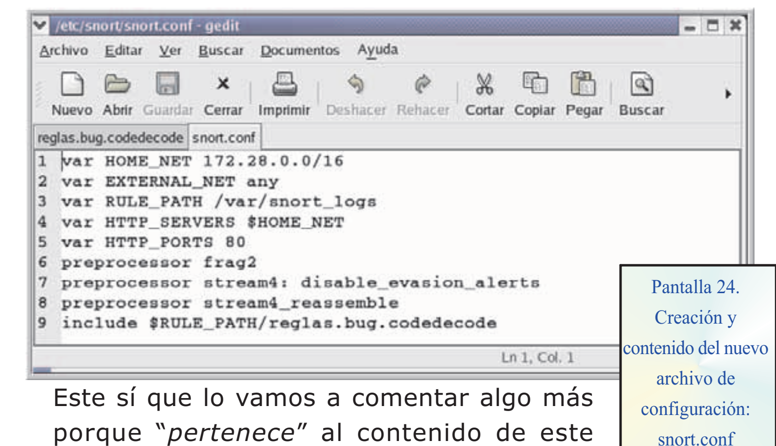


Pantalla 23. Cambiar de nombre el archivo de configuración a snort.conf.old

Con esto conseguiremos no perder el archivo original

Paso 3º) Crear el nuevo archivo de configuración en el directorio de ejecución del programa... para *LiNUX* el lugar será ***/etc/snort*** y para *Windows* será ***C:\snort\etc*** .

El archivo a crear se llamará ***snort.conf*** y tiene este contenido:



Pantalla 24. Creación y contenido del nuevo archivo de configuración: snort.conf

Este sí que lo vamos a comentar algo más porque *"pertenece"* al contenido de este artículo:

Línea 1 indica la **red en la que estamos** (***172.28.16.0.0/16***)

Línea 2, la **red externa**... cualquiera (***any***)

Línea 3, el **camino que apunta al archivo o archivos de reglas** (este debe ser el mismo directorio en donde guardamos las reglas establecidas en el paso 1º)

Línea 4 y línea 5, indican la **dirección IP y puerto de nuestro servidor web**, observa que se puso ***$HOME_NET*** como dirección IP. Es decir, ***snort*** monitorizará cualquier servidor web de la red 172.28.0.0 /16

Líneas 6, 7 y 8, incluyen las **directivas del preprocesador**, sólo hacemos referencia a ***frag2*** y ***stream4,*** recuerda lo que se dijo al principio de ellos y para lo que servían.

Línea 9, indica cual será el o los **archivos de reglas a aplicar**, cada archivo de reglas debe nombrarse con un ***include*** por archivo, en nuestro caso, sólo hay un archivo de reglas.

Paso 4º) Ejecutaremos ***snort*** indicándole la forma de aplicar esas reglas...

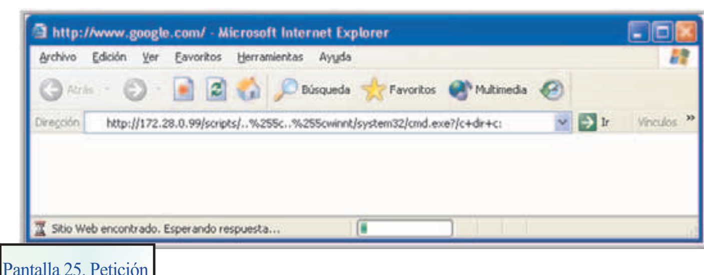
snort –de –l /var/snort_logs –c /etc/snort/snort.conf

La opción –l le sigue un directorio dónde se guardarán los ***logs***, como en la práctica número 3 pero a diferencia de ella, ahora sólo aplicará las reglas establecidas y por tanto sólo registrará aquello que tenga que ver con esas reglas, además cuando se descubra una alerta creará un archivo de texto llamado ***alerts*** que guardará los registros de las mismas. (si lo pruebas en plataformas Windows, ese archivo se llamará ***alerts.ids***

Otra diferencia es que omitimos el parámetro –v y no se mostrará nada por pantalla, luego te toca practicar en casa con algunas de las opciones que vimos... es mejor no indicar la opción ***verbose (-v)*** puesto que así no cargamos de trabajo extra a ***snort*** y a nuestra máquina...

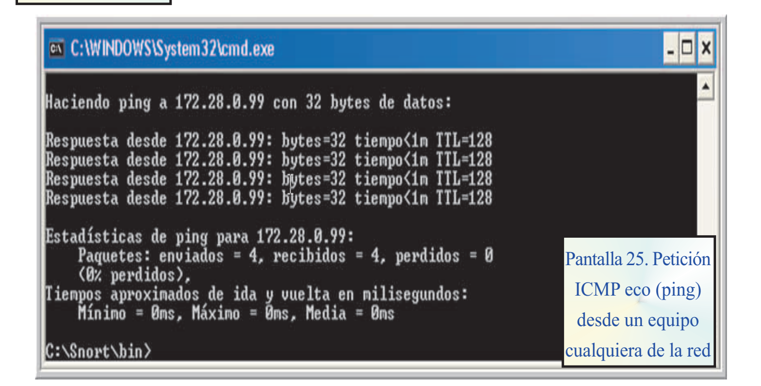
Paso 5º) Accesos

Sólo nos queda probar a ver que pasa... así que vamos a coger otro equipo y "*navegaremos*" hacia el servidor web intentando un acceso por *unicode*... luego veremos que ocurrió:



Pantalla 25. Petición web para explotar el bug de unicode desde un equipo cualquiera de la red

Desde otro equipo o desde el mismo hacemos un ping para probar la regla de ICMP....

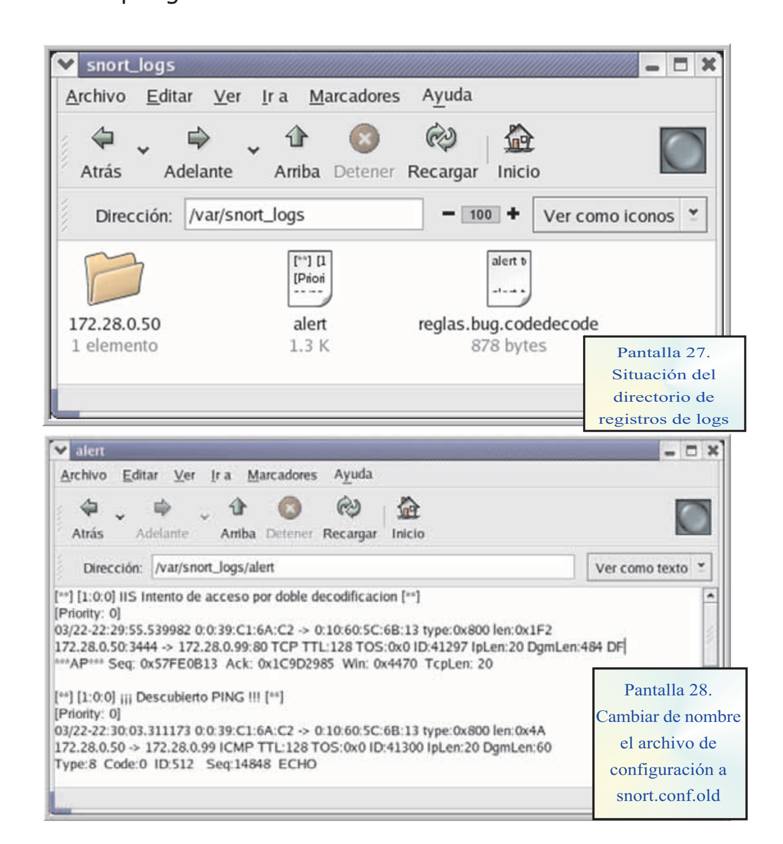


Pantalla 25. Petición ICMP eco (ping) desde un equipo cualquiera de la red

Paso 6º) Situación de alertas y logs

Veamos que ocurrió en el directorio ***/var/snort_logs*** donde está ejecutándose el IDS..

Lo primero que vemos es que se crearon **una carpeta** (***172.28.0.50***, que corresponde a la IP del equipo "*atacante*") **y un archivo alert** que guarda las alarmas.



Pantalla 27. Situación del directorio de registros de logs

Pantalla 28. Cambiar de nombre el archivo de configuración a snort.conf.old

Como verás aparecen las alertas del intento de explotación tal y como definimos en las reglas y también el/los *pings* realizados contra el servidor web

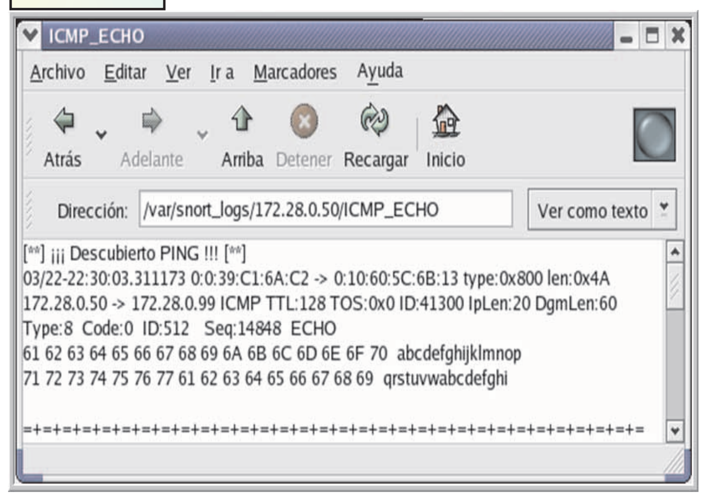


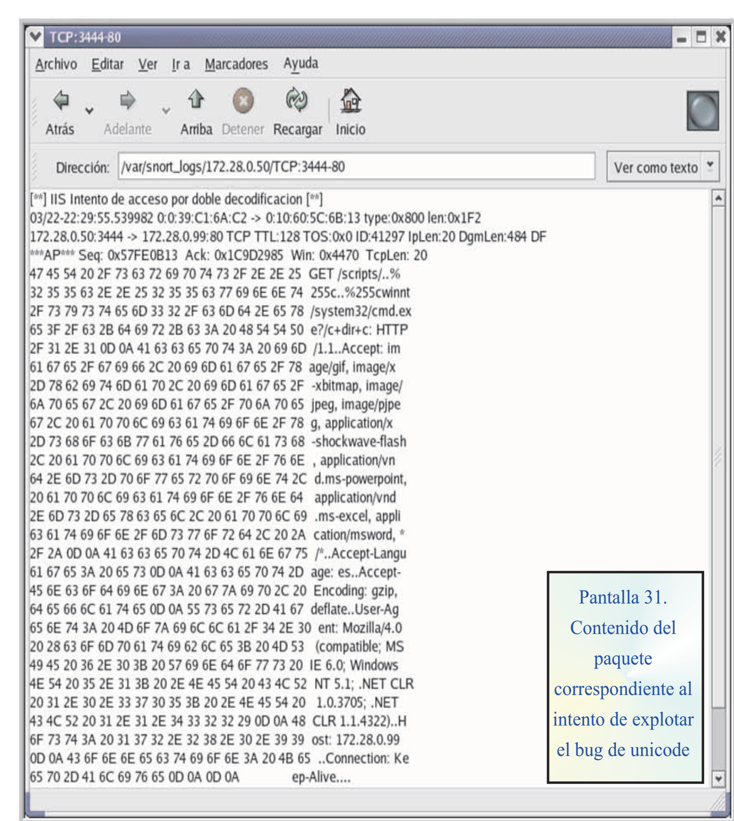
Pantalla 29 Contenido de la carpeta de IP's logeadas

Los dos archivos que contiene esta carpeta guardan información detallada acerca de cada una de las acciones, ***ICMP_ECHO*** se **corresponde con el *ping*** y **TCP:3444-80 es la información detallada del intento por *unicode***

Así que terminemos el artículo y veamos el contenido de ambos:

Pantalla 30. Contenido del paquete correspondiente a la petición ping





Pantalla 31. Contenido del paquete correspondiente al intento de explotar el bug de unicode

Ya sé que ha sido *"excesiva"* la profusión de pantallas, comentarios y otras *"lindezas"* pero es mejor ahora que luego... y en estos casos: *mejor pecar en exceso que quedarse con mil dudas*, en los próximos números no será *"tan minucioso"* el análisis, puesto que ya tendremos cierta experiencia y además habrás tenido mucho más tiempo para practicar con ***snort***.

Espero que todo se haya entendido correctamente, nos queda un largo trecho y la parte más complicada de ***snort***, definir reglas, contramedidas ante los ataques, informes, plug-ins de salida, etc.. todo ello en su conjunto hará nuestra vida por la red más segura y si después de todo ello implementamos un *firewall* en condiciones, será bastante difícil que nos invadan las *malas compañías.*

Un saludo.

SERVIDOR DE HXC
MODO DE EMPLEO

- Hack x Crack ha habilitado tres servidores para que puedas realizar las prácticas de hacking.

- Las IPs de los servidores de hacking las encontrarás en EL FORO de la revista (www.hackxcrack.com). Una vez en el foro entra en la zona COMUNICADOS DE HACK X CRACK (arriba del todo) y verás varios comunicados relacionados con los servidores. No ponemos las IP aquí porque es bueno acostumbrarte a entrar en el foro y leer los comunicados. Si hay alguna incidencia o cambio de IP o lo que sea, se comunicará en EL FORO.

- Actualmente tienen el BUG del Code / Decode. La forma de "explotar" este bug la explicamos extensamente en los números 2 y 3. Lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack.

- En los Servidores corre el Windows 2000 con el IIS de Servidor Web. No hemos parcheado ningún bug, ni tan siquiera el RPC y por supuesto tampoco hemos instalado ningún Service Pack. Para quien piense que eso es un error (lógico si tenemos en cuenta que el RPC provoca una caída completa del sistema), solo decirte que AZIMUT ha configurado un firewall desde cero que evita el bug del RPC, (bloqueo de los puertos 135 (tcp/udp), 137 (udp), 138 (udp), 445 (tcp), 593 (tcp)). La intención de todo esto es, precisamente, que puedas practicar tanto con el CODE/DECODE como con cualquier otro "bug" que conozcas (y hay cientos!!!). Poco a poco iremos cambiando la configuración en función de la experiencia, la idea es tener los Servidores lo menos parcheados posibles pero mantenerlos operativos las 24 horas del día. Por todo ello y debido a posibles cambios de configuración, no olvides visitar el foro (Zona Comunicados) antes de "penetrar" en nuestros servidores.

- Cada Servidor tiene dos unidades (discos duros duros):
* La unidad c: --> Con 40GB y Raíz del Sistema
* La unidad d: --> Con 40GB
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad c:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en c:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en c:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (c:\inetpub) como su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web (IIS) y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de uno de nuestros servidores) en tu navegador (el Internet explorer por ejemplo), lo que estás haciendo realmente es ir al directorio c:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html ... pero... ¿qué te estoy contando?... si has seguido nuestra revista ya dominas de sobras el APACHE ;)

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco… … ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hackeado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser posible y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :) ... ... ¿cómo? ¿que no sabes crear directorios mediante el CODE/DECODE BUG... repasa los números 2 y tres de Hack x Crack ;p

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\xxx\system32\default\10019901\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (esperemos que muchos años).

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

CURSO DE PHP

APRENDE A MANEJAR SOCKETS CON PHP

- Aprenderemos a manejar SOCKETS
- Tocaremos algunos conceptos de TCP / IP
- Programaremos un Servidor PHP
- Qué es un SOCKET
- Programaremos un Cliente en PHP

Continuamos con el curso de PHP, en este número aprenderemos a manejar sockets. La programación de socket da mucho juego, pero es necesario conocer conceptos básicos de TCP/IP. Con socket se puede programar un servidor, un cliente telnet, un servidor de FTP, ... un sin fin de aplicaciones. Este capítulo es tal vez el más complejo ya que hace falta conocer TCP/IP y conocimientos de redes. Daremos varios ejemplos de cómo programar un servidor y un cliente.

Un poco de culturilla sobre el origen de los sockets.

No desesperes si no entiendes muchos de los conceptos que estás a punto de leer en esta introducción, ¿vale?, **al final verás un cuadro de resumen con lo verdaderamente importante**.

Unix fue desarrollado a finales de los años sesenta y principios de los setenta, y se diseñó originalmente como un sistema orientado al proceso, en el que cada programa de aplicación se ejecuta como un proceso de nivel de usuario. Un programa de aplicación interactúa con el sistema operativo haciendo llamadas de sistema.

Desde el punto de vista del programador, las llamadas de sistema se ven y comportan exactamente igual que las demás llamadas de procedimiento. Toman argumentos y devuelven uno o más resultados. Los argumentos pueden ser valores (por ejemplo, una operación de enteros) o punteros a objetos en el programa de aplicación (como un búfer que ha de ser llenado con caracteres).

Derivados de los Multics y los sistemas anteriores, los sistemas primitivos de entrada y salida de UNIX siguen un paradigma que algunas veces se denomina open-read-write-close.

- Antes de que un proceso de usuario pueda ejecutar operaciones de E/S, llama a open para especificar el archivo o dispositivo que va a usar y obtiene el permiso.
- La llamada a open devuelve un pequeño entero descriptor de archivo, que el proceso utiliza cuando ejecuta las operaciones de E/S en el archivo o dispositivo abierto.
- Una vez que se ha abierto un objeto, el proceso de usuario hace una o más llamadas a read o write para transferir datos.
- Read transfiere datos dentro del proceso de usuario.
- Write transfiere datos del proceso de usuario al archivo o dispositivo.
- Tanto read como write toman tres argumentos que especifican el descriptor de archivo que se ha de usar, la dirección de un búfer y el número de octetos que se han de transferir.
- Luego de completar todas las operaciones de transferencia, el proceso de usuario llama a close para informar al sistema operativo que ha terminado de usar el objeto.

Originalmente, los diseñadores de UNIX agruparon todas las operaciones de E/S en el paradigma open-read-write-close descrito

anteriormente. EL esquema incluyó E/S para los dispositivos orientados a caracteres (como los teclados) y para dispositivos orientados a bloques (como los discos y los archivos de datos).

Una de las primeras implantaciones del TCP/IP bajo Unix también utilizó el paradigma open-read-write-close con un nombre de archivo especial, /dev/tcp

El grupo que añadió los protocolos de red a BSD de UNIX decidió que, como los protocolos de la red eran más complejos que los dispositivos convencionales de E/S, la interacción entre los procesos del usuario y los protocolos de red debía ser más compleja que las interacciones entre los procesos de usuario y las instalaciones convencionales de E/S.

En particular, la interfaz de protocolo de debía permitir a los programadores crear un código de servidor que esperase las conexiones pasivamente, así como también un código cliente que formara activamente las conexiones.

Además, los programas de aplicación que mandaban datagramas podían especificar la dirección de destino junto con cada datagrama en lugar de destinos enlazados en el momento en que llamaban a open.

Para manejar todos estos casos, los diseñadores eligieron abandonar el paradigma tradicional de UNIX open-read-write-close y añadir varias llamadas nuevas del sistema operativo, así como una nueva biblioteca de rutinas.

Posteriormente surgió una mayor complejidad en la interfaz de protocolos de UNIX pues los diseñadores intentaron construir un mecanismo general para adaptar muchos protocolos. Por ejemplo, la generalidad hace posible, para el sistema operativo, incluir software para otros conjuntos de protocolos así como también TCP/IP y permitir que un programa de aplicación utilice uno o más de ellos a la vez. Como consecuencia, el programa de aplicación no sólo puede proporcionar una dirección de 32 bits y esperar a que el sistema operativo la interprete de manera correcta, la aplicación debe especificar explícitamente que el número de 32 bits representa una dirección IP.

La base para la E/S de la red en BSD de UNIX se centra en una abstracción conocida como socket. Imagina el socket como una generalización del mecanismo de acceso a archivos de UNIX que proporciona un punto final para la comunicación.

Al igual que con el acceso a archivos, los programas de aplicación requieren que el sistema operativo cree un socket cuando se necesita. El sistema devuelve un número entero pequeño que utiliza el programa de aplicación para hacer referencia al socket recientemente creado.

La diferencia principal entre los descriptores de archivos y los descriptores de socket es el sistema operativo enlaza un descriptor de archivo a un archivo o dispositivo específico cuando la aplicación llama a open, pero puede crear sockets sin enlazarlos a direcciones de destino específicas.

La aplicación puede elegir abastecer una dirección de destino cada vez que utiliza el socket (es decir, cuando se envían datagramas) o elegir enlazar la dirección de destino a un socket y evadir la especificación de destino repetidamente (es decir, cuando se hace una conexión TCP).



CUADRO RESUMEN

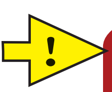
CUADRO DE RESUMEN: La manera de trabajar con socket es similar a los ficheros. Los socket nos permiten conectar con otras máquinas. Es importante saber que todos los servidores TCP/IP funcionan mediante socket, es decir, que si comprendes el paradigma socket estarás comprendiendo el verdadero corazón de todas las aplicaciones de Internet TCP/IP (Telnet, FTP, email, ...)

Todo eso está muy bien, ¿pero que es realmente un socket?

La mayoría de las aplicaciones que funcionan en un entorno de red están formadas por dos elementos: el **cliente** y el **servidor**. Para poder enviar órdenes a un servidor que está atendiendo nuestras peticiones se utilizan los **sockets**.

Para definir qué es un **socket**, pensemos en ello como una pequeña tubería por la que se pueden enviar datos.

Como existen diferentes tipos de datos a enviar en función del **servicio** utilizado (Web, FTP, e-mail, Telnet...) existe una forma de definir qué "tubería" queremos utilizar: se define mediante un número que se denomina **puerto**. De esta forma se puede acceder a un **servidor** que mantenga activos diferentes **servicios** y, por lo tanto, se acceda a través de diferentes **puertos** a un **servicio** concreto.



El paralelismo

EL PARALELISMO: Tuberías de líquidos.

Para quien todavía esté perdido entre palabras como cliente, servidor, sockets, puertos y servicios, vamos a hacer un paralelismo con la vida real.

Imaginemos que una de esas "súper-multinacionales" (tan de moda hoy en día) monta una fábrica en Alemania que fabrica (sirve, servidor) unos 65.000 tipos de líquidos y otra en España que consume (cliente) esos líquidos.

Como son muy "chulos" y no quieren depender de nadie, para trasladar los líquidos de Alemania a España montan tubería MUY MUY LARGA y MUY MUY GRUESA que va desde la fábrica de Alemania hasta la fábrica de España (pidiendo los permisos pertinentes a dichos países, claro ;)

Imagina que desde Alemania quieren enviar a España gasolina y coca-cola. Tienen un problema: primero tendrán que enviar coca-cola, después limpiar la súper-tubería, enviar el gas-oil y volverla a limpiar porque mañana pueden querer enviar, vete a saber, pesticida para los cultivos de arroz (no sería una buena idea juntar pesticida con coca-cola, ¿verdad?).

Para solucionar el enorme gasto de tiempo y dinero que supone limpiar la súper-tubería cada vez que cambian de líquido, deciden meter dentro de la súper-tubería unas 65000 mangueras. Y deciden que, a partir de ahora, por la manguera 1 (puerto 1) enviarán gasolina, por la manguera 2 (puerto 2) coca-cola, por la manguera tres (puerto 3) acido acético y así hasta las 65000.

Por otro lado, cada extremo de cada manguera se conecta a un grifo (socket). Es decir, la manguera 1 (puerto 1) tiene un extremo conectado a un grifo en Alemania (socket) y otro extremo conectado a un grifo (socket) en España.

VAMOS A INTENTARLO!!!
La fábrica alemana (servidor) quiere enviar a la fábrica Española (cliente) un líquido (datos) y para ello abre el grifo (socket) de la manguera 80 (puerto 80). El líquido (datos) pasa por la manguera 80 (puerto 80) y llega hasta el otro extremo, hasta el grifo (socket) que hay en España.

Hemos dicho que cada manguera debía contener un producto, y en este caso los ejecutivos de la multinacional decidieron que "normalmente" (ya sabes que los ejecutivos pueden cambiar de opinión) por la manguera 80 (puerto 80) circulará líquido (datos) del tipo "agua mentolada" (Web). Por lo tanto, a eso lo llaman servicio de "agua mentolada" (servicio Web).

La verdad es que se lo han montado muy bien, la fábrica alemana (servidor) puede servir a la fábrica Española (cliente) hasta 65.000 tipos de líquidos (65000 servicios, Servicios Web, Servicios FTP, Servicios...).

No quiero extenderme con el paralelismo porque hay pocas páginas en la revista, solo imagina que los ejecutivos se dan cuenta que los líquidos (datos) pueden ir tanto de Alemania a España como de España a Alemania (un cliente se puede transformar en servidor y un servidor en cliente). Además, los paralelismos son simples aproximaciones (a veces demasiado simples), a medida que aprendas sobre un tema tendrás que "adaptar" (y ampliar) este paralelismo.

Hay dos tipos de estudiantes (y todos somos alumnos de la vida), los que simplemente acumulan datos (enciclopedias andantes) y los que transforman los datos en conocimiento (los que entienden)... ya sabes a quien va dirigida esta revista ;)

PHP proporciona acceso a lo que se denomina socket de Berkeley. Se trata, por lo tanto, de un conjunto de funciones (de nuevo una API) que permite establecer una comunicación con otro sistema que puede recibir y enviar información utilizando esta técnica.

Un socket puede funcionar como servidor o como cliente (espero que después del paralelismo “Tuberías de líquidos” entiendas esta frase :)

Un socket de servidor es un servicio al que se le ha asignado un puerto por el cual escucha y responde. Por ejemplo, si abres varios navegadores Web y visitas páginas diferentes, estarás abriendo varios socket diferentes y todo ellos tienen asignados el mismo puerto (puerto 80). Esto sería como una manguera que en su extremo tiene varios grifos ;)

Los servidores de emails (POP3, SMTP) y servidores Web son buenos ejemplos de servidores sockets. El servidor HTTP (Web) escucha por el puerto 80 las solicitudes y responde con información HTML y otros ficheros (por ejemplo imágenesimagenes).

Generalmente el socket servidor se encuentra continuamente funcionando, por lo general es un servicio de Windows o un demonio de Linux.

Tipos de socket

Si estás siguiendo el curso de TCP/IP ya sabrás que hay dos formas de enviar la información, por TCP y UDP, esto quiere decir que la información por Internet puede ser enviada por uno de estos dos protocolos. La información es dividida en paquetes y luego ensamblada en el destino.

Si hay dos formas de enviar información, entonces PHP tiene que soportar la creación de sockets tanto TCP como UDP.

¿Es importante saber esto de TCP y UDP?, si que lo es, ya que es necesario para saber como conectar con servidores y obtener información ....

TCP (Transmission Control Protocol), la transmisión de los paquetes está numerada y es ensamblada en el destino siguiendo la numeración de los paquetes. Si un paquete falta se vuelve a solicitar. Es decir, TCP es un protocolo seguro que no permite perdida de datos.
Ejemplos de protocolos TCP: HTTP, FTP, EMAIL, TELNET, ...

UDP (User Datagram Protocol), este protocolo permite perder datos, si algún paquete se pierde no se vuelve a solicitar, se asume la perdida, ¿qué servicios pueden utilizar UDP?, pues por ejemplo los protocolos de video por Internet, de voz, música por Internet, donde no es crucial el 100% la calidad y donde se solicita información en tiempo real, lógicamente el UDP es más rápido que el TCP.

PHP Socket

PHP soporta el manejo de sockets a bajo nivel. En PHP3 se introdujo manejar socket gracias a la función fosckopen() entre otras funciones asociadas.

Las funciones socket en PHP están en fase experimental y aún pueden ser modificadas, pero lo cierto es que funcionan muy bien y se puede crear aplicaciones con estas funciones.

Creando socket en PHP

Programar socket en PHP es muy similar a cómo se programa con funciones en C. Vamos a comenzar con un simple ejemplo: un Servidor socket que escucha una conexión por el puerto 9000, acepta una cadena de texto como entrada y retorna la misma cadena de texto pero borrando los espacios en blanco.

```
<?php
// Tiempo ilimitado de ejecución, para que esté funcionando eternamente.

set_time_limit (0);

// La IP y el puerto por donde van a escuchar, recuerda que una misma
// máquina puede tener varias IPs
$address = ‘127.0.0.1’;
$port = 9000;

// Crea un socket TCP Stream
$sock = socket_create(AF_INET, SOCK_STREAM, 0);
// Se asocia el socket a la direccion IP y al puerto
```

```
socket_bind($sock, $address, $port) or die('Error al asociar');
// Escucha por el socket
socket_listen($sock);

/* Se comunica que acepte la conexión */
$client = socket_accept($sock);

// Lee 1024 caracteres de la conexión
$input = socket_read($client, 1024);

// Reemplaza los espacio en blanco
$output = ereg replace("[ \t\n\r]","",$input).chr(0);

// Envía la cadena al cliente
socket_write($client, $output);

// Cierra la conexión
socket_close($client);

// Cierra el socket
socket_close($sock);
?>
```

Vamos a comentar el programa. Recuerda que el cometido del programa es hacer de servidor, de recibir un texto y luego devolverlo sin espacios en blanco (y todo ello en TCP). Un Servidor Web funciona de forma muy parecida, pero sirviendo páginas HTML.

1. $sock = socket_create(AF_INET, SOCK_STREAM, 0);

Se crea un socket que hará de servidor, para cliente ya lo veremos después.

- AF_INET indica que el protocolo a aceptar será IPV4, usado tanto para TCP o UDP.
- SOCK_STREAM indica que será un socket full duplex, aceptando peticiones de tipo streams.
- Resumiendo: Con esta línea indicamos que se cree un socket en TCP. Si fuera UDP habría sustituir SOCK_STREAM por SOCK_DGRAM.

2. socket_bind($sock, $address, $port) or die('Error al asociar');

Una vez que ya está creado el socket es necesario asociarlo a una IP y aun puerto para que sepa por donde trabajar.

3. socket_listen($sock);

Escucha por el socket, es decir, va a esperar a que llegue información por el puerto 9000 de la IP 127.0.0.1

4. $client = socket_accept($sock);

Al ser TCP, el cliente solicita una conexión. Esta línea hace que el Servidor acepte la conexión, de esta forma podrá capturar la información procedente del cliente mediante el socket.

5. $input = socket_read($client, 1024);

Una vez que la conexión TCP ha sido aceptada se captura 1024 caracteres o hasta que se envíe un retorno de carro (pulsando la tecla Return) enviados por el cliente.

6. $output = ereg replace("[\t\n\r]","",$input).chr(0);

La función ereg_replace ya se ha comentado en anteriores capítulos, en este ejemplo elimina los espacios en blanco.

7. socket_write($client, $output);

Escribe el resultado en la conexión establecida, al cliente le llegará el texto enviado pero sin espacios.

8. Socket_close($cliente) y socket_close($sock)

Cierra la conexión y el socket, para liberar recursos.

Creando un servidor real

Ahora que ya conocemos los conceptos básicos para trabajar con socket, vamos a crear un verdadero servidor. El anterior ejemplo tenía la limitación de que solo aceptaba una

conexión, es como si un Servidor Web solo aceptara una visita, lógicamente eso no puede decirse que sea un servidor de verdad.

El siguiente ejemplo aceptará varias conexiones de varios clientes, ¡¡esto si que es un verdadero servidor!!.

Para conseguir el que programa nunca termine y siempre acepte nuevas conexiones podemos usar while(true) { /* código */ }.

El siguiente ejemplo es similar al anterior pero con algunas nuevas funcionalidades:

- El programa no termina al aceptar una conexión.
- Crea un sistema fácil para finalizar el servidor.
- Acepta simultáneamente varios clientes.

```
<?php
// Tiempo de ejecución ilimitado
set_time_limit (0);

// IP y puerto de escucha
$address = '127.0.0.1';
$port = 9000;
$max_clients = 10;

// Crea un array de clientes (para aceptar varias peticiones)
$clients = Array();

// Crea un socket TCP
$sock = socket_create(AF_INET, SOCK_STREAM, 0);
// Asocia el socket al puerto y dirección
socket_bind($sock, $address, $port) or die('Error al asociar');
// Comienza a leer por el socket
socket_listen($sock);

// Ciclo infinito, para que no termine de aceptar peticiones
while (true) {
// Inicia el array para aceptar conexiones
$read[0] = $sock;
for ($i = 0; $i < $max_clients; $i++)
{
if ($client[$i]['sock'] != null)
$read[$i + 1] = $client[$i]['sock'] ;
}
// Set up a blocking call to socket_select()
$ready = socket_select($read,null,null,null);
/* Si hay una nueva conexión lo añade al array de conexiones */
if (in array($sock, $read)) {
for ($i = 0; $i < $max_clients; $i++)
{
if ($client[$i]['sock'] == null) {
$client[$i]['sock'] = socket_accept($sock);
break;
}
elseif ($i == $max_clients - 1)
print ("Demasiados clientes")
}
if (--$ready <= 0)
continue;
} // Fin del array de clientes
// Si el cliente está intentando leer le asocia un handle y lee el contenido

for ($i = 0; $i < $max_clients; $i++)
{
if (in array($client[$i]['sock'] , $read))
{
$input = socket_read($client[$i]['sock'] , 1024);
if ($input == null) {
// Si la longitud es cero desconecta
unset($client[$i]);
}
$n = trim($input);
if ($input == 'exit') {
// Si se recibe la palabra exit se desconecta
socket_close($client[$i]['sock']);
} elseif ($input) {
// Elimina los espacios en blanco
$output = ereg replace("[ \t\n\r]","",$input).chr(0);
socket_write($client[$i]['sock'],$output);
}
} else {
// Cierra los socket
socket_close($client[$i]['sock']);
unset($client[$i]);
}
}
}
socket_close($sock);
?>
```

La funcionalidad del programa es la misma del primer ejemplo, con algunas mejoras – cuando el cliente envía la palabra EXIT el servidor finaliza la conexión con ese usuario.

El programa es muy similar al primero, excepto el bucle infinito y algunos bloques de código:

1. Inicia un socket para cliente.
2. Escucha a cada cliente y lo inicia en el array $client.
3. Atiende a cada cliente, recibe y envía los datos.
4. Un handle para cada cliente.

Ejemplo de cliente Socket

Programar un cliente socket es mucho más sencillo que programar un servidor. El siguiente ejemplo se conecta a www.hackxcrack.com, envía una petición HEAD, imprime la respuesta y sale.

```
<?php
error_reporting(E_ALL); // Imprime en pantalla los errores que ocurran

echo "<h2>Conexi&oacute;n TCP/IP</h2>\n";

/* Obtener el puerto para el servicio WWW. */
$puerto_servicio = getservbyname('www', 'tcp');

/* Obtener la direccion IP del host de destino. */
$direccion = gethostbyname('www.hackxcrack.com');

/* Crear un socket TCP/IP. */
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if ($socket < 0) {
echo "socket_create() fall&oacute;: motivo: " . socket_strerror($socket) . "\n";
}
else {
echo "OK.\n";
}

echo "Intentando una conexi&oacute;n con '$direccion' en el puerto '$puerto_servicio'...";
$resultado = socket_connect($socket, $direccion, $puerto_servicio);
if ($resultado < 0) {
echo "socket_connect() fall&oacute;.\nMotivo: ($resultado) " .
socket_strerror($resultado) . "\n";
}
else {
echo "OK.\n";
}
$entrada = "HEAD / HTTP/1.1\r\n";
$entrada .= "Host: www.hackxcrack.com\r\n";
$entrada .= "Connection: Close\r\n\r\n";
$salida = '';
echo "Enviando petici&oacute;n HTTP HEAD...";
socket_write($socket, $entrada, strlen($entrada));
echo "OK.\n";
echo "Leyendo respuesta:\n\n";
while ($salida = socket_read($socket, 2048)) {
echo $salida;
}
echo "Cerrando socket...";
socket_close($socket);
echo "OK.\n\n";
?>
```

Listado de funciones principales para programar socket

Socket_Accep()

Acepta una conexión. Esta función es experimental, aún está en fase de desarrollo y puede variar en futuras versiones de PHP.

```
<?php
$fd = socket_create(AF_INET, SOCK_STREAM, 6);
$PORT = 5000;
socket_bind($fd, "0.0.0.0", $PORT);
while(true)
{
$remote_fd = socket_accept($fd);
remote_socket_client_handle($remote_fd);
}  ?>
```

socket_create (int domain, int type, int protocol)

Crea un socket y retorna un identificador del socket.

Domain toma los siguientes valores:

Domain	Descripción
AF_INET	Protocolo IPv4. TCP y UDP utilizan esta familia de protocolos.
AF_INET6	Protocolo Ipv6. TCP y UDP utilizan esta familia de protocolos. Solo disponible en PHP 5.0
AF_UNIX	Protocolo de comunicación local.

Type	Descripción
SOCK_STREAM	Conexión segura y full duplex. Es el tipo utilizado por el protocolo TCP.
SOCK_DGRAM	Suporta datagramas, conexión no segura. Es el tipo utilizado por el protocolo UDP.

Protocol	Descripción
icmp	Internet Control Message Protocol es un protocolo principalmente utilizado por los Gateways y hosts para comprobar el estado de estos, generando reportes de errores. El comando PING es un ejemplo de ICMP.
udp	Protocolo basado en datagramas. No es un protocolo seguro.
tcp	Protocolo basado en paquetes, full duplex y seguro. La mayoría de los servicios en internet funcionen con tcp.

bool socket_bind (resource socket, string address [, int port])

Asocia un socket a un puerto y una dirección IP. La función es aún experimental por lo que puede sufrir modificaciones.

Socket toma el valor del puntero que se a creado con la función socket_create(). El parámetro address es una dirección IP (ejemplo: 127.0.0.1). El parámetro port es solo usado cuando la conexión es de tipo AF_INET, y asigna el puerto al que se debe conectar la aplicación.

Devuelve True si todo se llevó correctamente, FALSE en caso contrario.

bool socket_listen (resource socket [, int backlog])

Después de que el socket haya sido creado con socket_create() y asociado a una IP/puerto con la función socket_bind(), se necesita que el socket esté escuchando esperando conexiones, para ello se utiliza la función socket_listen.

string socket_read (resource socket, int length [, int type])

La función socket_read() lee desde un socket creado con socket_create() o socket_accept(). El número máximo de bytes a leer es especificado por el parámetro length.

Socket_read() retorna los datos como una cadena de texto, o texto si se ha producido un error.
Es importante saber que socket_read() debe retornar una cadena de tamaño 0 para indicar

En el próximo número:
Programar socket te puede resultar algo complicado, la mejor manera de aprender es experimentando, juega con las funciones principales de sockets, observa los errores, crea servidores TCP y luego adáptalos a UDP.
En el próximo número seguiremos con socket y comenzaremos a tratar fechas en PHP.

David C.M



En la editorial...

En la editorial de la revista siguen llegando mails preguntando como se ejecuta el código PHP. Hemos dedicado números enteros a explicarlo, desde cómo montar el Servidor Apache hasta la implementación de PHP (pasando por todo tipo de configuraciones). Revisa los números anteriores y si te quedan dudas pregunta en el foro de hackxcrack (www.hackxcrack.com).

SUSCRIBETE A PC PASO A PASO

SUSCRIPCIÓN POR:
1 AÑO
11 NUMEROS

=

45 EUROS (10% DE DESCUENTO)
+
SORTEO DE UNA CONSOLA XBOX
+
SORTEO 2 JUEGOS PC (A ELEGIR)

Contra Reembolso

Solo tienes que enviarnos un mail a preferente@hackxcrack.com indicando:
- Nombre
- Apellidos
- Dirección Completa
- Población
- Provincia
- Cógigo Postal
- Mail de Contacto y/o Teléfono Contacto
Es imprescindible que nos facilites un mail o teléfono de contacto.
- Tipo de Subscripción: CONTRAREEMBOLSO
- Número de Revista:
Este será el número a partir del cual quieres subscribirte. Si deseas (por ejemplo) subscribirte a partir del número 5 (incluido), debes poner un 5 y te enviaremos desde el 5 hasta el 15 (ambos incluidos)

APRECIACIONES:
* Junto con el primer número recibirás el abono de 45 euros, precio de la subscripción por 11 números (un año) y una carta donde se te indicará tu número de Cliente Preferente y justificante/factura de la subscripción.
* Puedes hacernos llegar estos datos POR MAIL,tal como te hemos indicado; rellenando el formulario de nuestra WEB (www.hackxcrack.com) o enviándonos una carta a la siguiente dirección:
CALLE PERE MARTELL Nº20, 2º-1ª
CP 43001 TARRAGONA
ESPAÑA
* Cualquier consulta referente a las subscripciones puedes enviarla por mail a preferente@hackxcrack.com

Giro Postal

Envíanos un GIRO POSTAL por valor de 45 EUROS a:
CALLE PERE MARTELL20, 2º 1ª.
CP 43001 TARRAGONA
ESPAÑA
IMPORTANTE: En el TEXTO DEL GIRO escribe un mail de contacto o un número de Teléfono.

Y enviarnos un mail a preferente@hackxcrack.com indicando:
- Nombre
- Apellidos
- Dirección Completa
- Población
- Provincia
- Cógigo Postal
- Mail de Contacto y/o Teléfono Contacto
Es imprescindible que nos facilites un mail o teléfono de contacto.
- Tipo de Subscripción: GIRO POSTAL
- Número de Revista:
Este será el número a partir del cual quieres subscribirte. Si deseas (por ejemplo) subscribirte a partir del número 5 (incluido), debes poner un 5 y te enviaremos desde el 5 hasta el 15 (ambos incluidos)

APRECIACIONES:
* Junto con el primer número recibirás una carta donde se te indicará tu número de Cliente Preferente y justificante/factura de la subscripción.
* Puedes hacernos llegar estos datos POR MAIL,tal como te hemos indicado; o enviándonos una carta a la siguiente dirección:
CALLE PERE MARTELL Nº20, 2º-1ª
CP 43001 TARRAGONA
ESPAÑA
* Cualquier consulta referente a las subscripciones puedes enviarla por mail a preferente@hackxcrack.com

CONSIGUE LOS NUMEROS ATRASADOS EN:

WWW.HACKXCRACK.COM



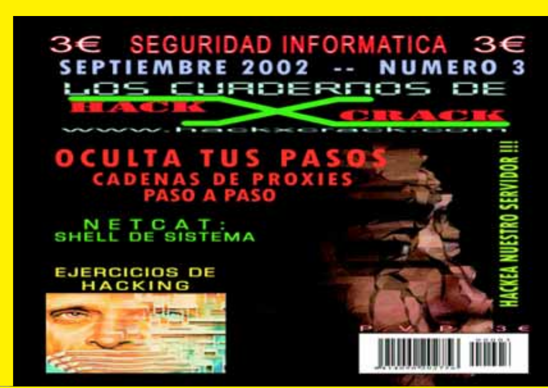
NÚMERO1:

-CREA TU PRIMER TROYANO
INDETECTABLE POR LOS ANTIVIRUS.
-FLASHFXP: SIN LÍMITE DE VELOCIDAD.
-FTP SIN SECRETOS: PASVMODE.
-PORT MODE/PASV MODE Y LOS FIREWALL: LA UTILIDAD DE LO APRENDIDO.
-TCP-IP:INICIACIÓN (PARTE 1).
-EL MEJOR GRUPO DE SERVIDORES FTP DE HABLA HISPANA.
-EDONKEY 2000 Y SPANISHARE.
-LA FLECHA ÁCIDA.



NÚMERO 2:

-CODE/DECODE BUG: INTRODUCCIÓN.
-CODE/DECODE BUG: LOCALIZACIÓN DEL OBJETIVO.
-CODE/DECODE BUG: LÍNEA DE COMANDOS.
-CODE/DECODE BUG: SUBIENDO ARCHIVOS AL SERVIDOR REMOTO.
-OCULTACIÓN DE IP: PRIMEROS PASOS.
-LA FLECHA ÁCIDA: LA SS DIGITAL.
AZNAR AL FRENTE DE LA SS DEL SIGLO XXI.



NÚMERO 3:

-PROXY: OCULTANDO NUESTRA IP.
ASUMIENDO CONCEPTOS.
-PROXY: OCULTANDO NUESTRA IP.
ENCADENANDO PROXIES.
-PROXY: OCULTANDO NUESTRA IP.
OCULTANDO TODOS NUESTROS PROGRAMAS TRAS LAS
CADENAS DE PROXIES.
-EL SERVIDOR DE HACKXCRACK: CONFIGURACIÓN Y MODO DE EMPLEO.
-SALA DE PRACTICAS: EXPLICACIÓN.
-PRÁCTICA 1ª: SUBIENDO UN ARCHIVO A NUESTRO SERVIDOR.
-PRÁCTICA 2ª: MONTANDO UN DUMP CON EL SERV-U.
-PRÁCTICA 3ª: CODE/DECODE BUG. LÍNEA DE COMANDOS.
-PREGUNTAS Y DUDAS.



NÚMERO 7:

- PROTOCOLOS: POP3
- PASA TUS PELICULAS A DIVX III (EL AUDIO)
- PASA TUS PELICULAS A DIVX IV (MULTIPLEXADO)
- CURSO DE VISUAL BASIC: LA CALCULADORA
- IPHXC: EL TERCER TROYANO DE HXC II
- APACHE: UN SERVIDOR WEB EN NUESTRO PC
- CCPROXY: IV TROYANO DE PC PASO A PASO
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 11:
- Curso de linux: programacion
- Visual Basic: IIS bug exploit
- Apache como proxy
- Serie Raw: FTP
- Validacion XML: DTD
- Historia: Lady Augusta Ada Byron



NÚMERO 12:
- Curso de linux: programacion C.
- Visual Basic: IIS bug exploit. Nuestro primer Scanner.
- APACHE: Configuralo de forma segura.
- Serie Raw: FTP(II)
- VALIDACION XML: DTD (II)



NÚMERO 13:
- Curso de linux: programacion C(II).
- Visual Basic:Nuestro primer proyecto.
- APACHE: Configuralo de forma segura.
- Serie Raw: HTTP.
- CURSO XML: DOM.



NÚMERO 15

- CURSO DE PHP (II)
- Xbox. Instalar Linux
- SERIE RAW (9): MSN
- CURSO VISUAL BASIC: UN CLIENTE, UNA NECESIDAD(III).
- PROGRAMACION BAJO LINUX: LENGUAJE C(III)



NÚMERO 17:
- Programación bajo linux: El sistema IPC(II)
- Curso de TCP/IP
- XBOX (III): Cambia el disco duro de tu XBOX
- Hackea Windows en 40 segundos
- Curso de PHP: Cadenas de texto

CONSIGUE LOS NUMEROS ATRASADOS EN:

WWW.HACKXCRACK.COM